                The RObust Header Compression (ROHC) Framework

Status of This Memo

   This document specifies an Internet standards track protocol for the
   Internet community, and requests discussion and suggestions for
   improvements. Please refer to the current edition of the "Internet
   Official Protocol Standards" (STD 1) for the standardization state
   and status of this protocol. Distribution of this memo is unlimited.

Copyright Notice

Abstract

   The Robust Header Compression (ROHC) protocol provides an efficient,
   flexible, and future-proof header compression concept. It is
   designed to operate efficiently and robustly over various link
   technologies with different characteristics.

   The ROHC framework, along with a set of compression profiles, was
   initially defined in RFC 3095. To improve and simplify the ROHC
   specifications, this document explicitly defines the ROHC framework
   and the profile for uncompressed separately. More specifically, the
   definition of the framework does not modify or update the definition
   of the framework specified by RFC 3095.

Table of Contents

1. Introduction

   For many types of networks, reducing the deployment and operational
   costs by improving the usage of the bandwidth resources is of vital
   importance. Header compression over a link is possible because some
   of the information carried within the header of a packet becomes
   compressible between packets belonging to the same flow.

   For links where the overhead of the IP header(s) is problematic, the
   total size of the header may be significant. Applications carrying
   data carried within RTP [13] will then, in addition to link-layer
   framing, have an IPv4 [10] header (20 octets), a UDP [12] header (8
   octets), and an RTP header (12 octets), for a total of 40 octets.
   With IPv6 [11], the IPv6 header is 40 octets for a total of 60
   octets. Applications transferring data using TCP [14] will have 20
   octets for the transport header, for a total size of 40 octets for
   IPv4 and 60 octets for IPv6.

   The relative gain for specific flows (or applications) depends on the
   size of the payload used in each packet. For applications such as
   Voice-over-IP, where the size of the payload containing coded speech
   can be as small as 15-20 octets, this gain will be quite significant.
   Similarly, relative gains for TCP flows carrying large payloads (such
   as file transfers) will be less than for flows carrying smaller
   payloads (such as application signaling, e.g., session initiation).

   As more and more wireless link technologies are being deployed to
   carry IP traffic, care must be taken to address the specific
   characteristics of these technologies within the header compression
   algorithms. Legacy header compression schemes, such as those defined
   in [16] and [17], have been shown to perform inadequately over links
   where both the lossy behavior and the round-trip times are non-
   negligible, such as those observed for example in wireless links and
   IP tunnels.

   In addition, a header compression scheme should handle the often
   non-trivial residual errors, i.e., where the lower layer may pass a
   packet that contains undetected bit errors to the decompressor. It
   should also handle loss and reordering before the compression point,
   as well as on the link between the compression and decompression
   points [7].

   The Robust Header Compression (ROHC) protocol provides an efficient,
   flexible, and future-proof header compression concept. It is
   designed to operate efficiently and robustly over various link
   technologies with different characteristics.

RFC 3095 [3] defines the ROHC framework along with an initial set of compression profiles. To improve and simplify the specification, the framework and the profiles' parts have been split into separate documents. This document explicitly defines the ROHC framework, but it does not modify or update the definition of the framework specified by RFC 3095; both documents can be used independently of each other. This also implies that implementations based on either definition will be compatible and interoperable with each other. However, it is the intent to let this specification replace RFC 3095 as the base specification for all profiles defined in the future.

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 [1].

2.1. Acronyms

This section lists most acronyms used for reference.

    ACK   Acknowledgment.
    CID   Context Identifier.
    CO   Compressed Packet Format.
    CRC   Cyclic Redundancy Check.
    IR   Initialization and Refresh.
    IR-DYN Initialization and Refresh, Dynamic part.
    LSB   Least Significant Bit(s).
    MRRU   Maximum Reconstructed Reception Unit.
    MSB   Most Significant Bit(s).
    MSN   Master Sequence Number.
    NACK   Negative Acknowledgment.
    ROHC   RObust Header Compression.

2.2. ROHC Terminology

Context

The context of the compressor is the state it uses to compress a header. The context of the decompressor is the state it uses to decompress a header. Either of these or the two in combination are usually referred to as "context", when it is clear which is intended. The context contains relevant information from previous headers in the packet flow, such as static fields and possible reference values for compression and decompression. Moreover, additional information describing the packet flow is also part of

the context, for example, information about the change behavior of
fields (e.g., the IP Identifier behavior, or the typical inter-
packet increase in sequence numbers and timestamps).

Context damage

When the context of the decompressor is not consistent with the
context of the compressor, decompression may fail to reproduce the
original header. This situation can occur when the context of the
decompressor has not been initialized properly or when packets
have been lost or damaged between the compressor and decompressor.

Packets which cannot be decompressed due to inconsistent contexts
are said to be lost due to context damage. Packets that are
decompressed but contain errors due to inconsistent contexts are
said to be damaged due to context damage.

Context repair mechanism

Context repair mechanisms are used to resynchronize the contexts,
an important task since context damage causes loss propagation.
Examples of such mechanisms are NACK-based mechanisms, and the
periodic refreshes of important context information, usually done
in unidirectional operation. There are also mechanisms that can
reduce the context inconsistency probability, for example,
repetition of the same type of information in multiple packets and
CRCs that protect context-updating information.

CRC-8 validation

The CRC-8 validation refers to the validation of the integrity
against bit error(s) in a received IR and IR-DYN header using the
8-bit CRC included in the IR/IR-DYN header.

CRC verification

The CRC verification refers to the verification of the result of a
decompression attempt using the 3-bit CRC or 7-bit CRC included in
the header of a compressed packet format.

Damage propagation

Delivery of incorrect decompressed headers due to context damage,
that is, due to errors in (i.e., loss of or damage to) previous
header(s) or feedback.

Error detection

   Detection of errors by lower layers. If error detection is not
   perfect, there will be residual errors.

Error propagation

   Damage propagation or loss propagation.

ROHC profile

   A ROHC profile is a compression protocol, which specifies how to
   compress specific header combinations. A ROHC profile may be
   tailored to handle a specific set of link characteristics, e.g.,
   loss characteristics, reordering between compression points, etc.
   ROHC profiles provide the details of the header compression
   framework defined in this document, and each compression profile
   is associated with a unique ROHC profile identifier [21]. When
   setting up a ROHC channel, the set of profiles supported by both
   endpoints of the channel is negotiated, and when initializing new
   contexts, a profile identifier from this negotiated set is used to
   associate each compression context with one specific profile.

Link

   A physical transmission path that constitutes a single IP hop.

Loss propagation

   Loss of headers, due to errors in (i.e., loss of or damage to)
   previous header(s) or feedback.

Packet flow

   A sequence of packets where the field values and change patterns
   of field values are such that the headers can be compressed using
   the same context.

Residual error

   Errors introduced during transmission and not detected by lower-
   layer error detection schemes.

ROHC channel

   A logical unidirectional point-to-point channel carrying ROHC
   packets from one compressor to one decompressor, optionally
   carrying ROHC feedback information on the behalf of another

compressor-decompressor pair operating on a separate ROHC channel
in the opposite direction. See also [5].

This document also makes use of the conceptual terminology defined by
"ROHC Terminology and Channel Mapping Examples", RFC 3759 [5].

3. Background (Informative)

This section provides a background to the subject of header
compression. The fundamental ideas are described together with a
discussion about the history of header compression schemes. The
motivations driving the development of the various schemes are
discussed and their drawbacks identified, thereby providing the
foundations for the design of the ROHC framework and profiles [3].

3.1. Header Compression Fundamentals

Header compression is possible because there is significant
redundancy between header fields; within the headers of a single
packet, but in particular between consecutive packets belonging to
the same flow. On the path end-to-end, the entire header information
is necessary for all packets in the flow, but over a single link,
some of this information becomes redundant and can be reduced, as
long as it is transparently recovered at the receiving end of the
link. The header size can be reduced by first sending field
information that is expected to remain static for (at least most of)
the lifetime of the packet flow. Further compression is achieved for
the fields carrying information that changes more dynamically by
using compression methods tailored to their respective assumed change
behavior.

To achieve compression and decompression, some necessary information
from past packets is maintained in a context. The compressor and the
decompressor update their respective contexts upon certain, not
necessarily synchronized, events. Impairment events may lead to
inconsistencies in the decompressor context (i.e., context damage),
which in turn may cause incorrect decompression. A Robust Header
Compression scheme needs mechanisms to minimize the possibility of
context damage, in combination with mechanisms for context repair.

3.2. A Short History of Header Compression

The first header compression scheme, compressed TCP (CTCP) [15], was
introduced by Van Jacobson. CTCP, also often referred to as VJ
compression, compresses the 40 octets of the TCP/IP header down to 4
octets. CTCP uses delta encoding for sequentially changing fields.
The CTCP compressor detects transport-level retransmissions and sends
a header that updates the entire context when they occur. This

repair mechanism does not require any explicit signaling between the compressor and decompressor.

A general IP header compression scheme, IP header compression [16], improves somewhat on CTCP. IP Header Compression (IPHC) can compress arbitrary IP, TCP, and UDP headers. When compressing non-TCP headers, IPHC does not use delta encoding and is robust. The repair mechanism of CTCP is augmented with negative acknowledgments, called CONTEXT_STATE messages, which speeds up the repair. This context repair mechanism is thus limited by the round-trip time of the link. IPHC does not compress RTP headers.

CRTP [17] is an RTP extension to IPHC. CRTP compresses the 40 octets of IPv4/UDP/RTP headers to a minimum of 2 octets when the UDP Checksum is not enabled. If the UDP Checksum is enabled, the minimum CRTP header is 4 octets.

On lossy links with long round-trip times, CRTP does not perform well [20]. Each packet lost over the link causes decompression of several subsequent packets to fail, because the context becomes invalidated during at least one link round-trip time from the lost packet. Unfortunately, the large headers that CRTP sends when updating the context waste additional bandwidth.

CRTP uses a local repair mechanism known as TWICE, which was introduced by IPHC. TWICE derives its name from the observation that when the flow of compressed packets is regular, the correct guess when one packet is lost between the compression points is to apply the update in the current packet twice. While TWICE improves CRTP performance significantly, [20] also found that even with TWICE, CRTP doubled the number of lost packets.

An enhanced variant of CRTP, called eCRTP [19], means to improve the robustness of CRTP in the presence of reordering and packet losses, while keeping the protocol almost unchanged from CRTP. As a result, eCRTP does provide better means to implement some degree of robustness, albeit at the expense of additional overhead, leading to a reduction in compression efficiency in comparison to CRTP.

4. Overview of Robust Header Compression (ROHC) (Informative)

4.1. General Principles

As mentioned earlier, header compression is possible per-link due to the fact that there is much redundancy between header field values within packets, and especially between consecutive packets belonging to the same flow. To utilize these properties for header compression, there are a few essential steps to consider.

The first step consists of identifying and grouping packets together
into different "flows", so that packet-to-packet redundancy is
maximized in order to improve the compression ratio. Grouping
packets into flows is usually based on source and destination host
(IP) addresses, transport protocol type (e.g., UDP or TCP), process
(port) numbers, and potentially additional unique application
identifiers, such as the synchronization source (SSRC) in RTP [13].
The compressor and decompressor each establish a context for the
packet flow and identify the context with a Context Identifier (CID)
included in each compressed header.

The second step is to understand the change patterns of the various
header fields. On a high level, header fields fall into one of the
following classes:

INFERRED   These fields contain values that can be inferred from
                other fields or external sources, for example, the size
                of the frame carrying the packet can often be derived
                from the link layer protocol, and thus does not have to
                be transmitted by the compression scheme.

STATIC   Fields classified as STATIC are assumed to be constant
                throughout the lifetime of the packet flow. The value
                of each field is thus only communicated initially.

STATIC-DEF   Fields classified as STATIC-DEF are used to define a
                packet flow as discussed above. Packets for which
                respective values of these fields differ are treated as
                belonging to different flows. These fields are in
                general compressed as STATIC fields.

STATIC-KNOWN   Fields classified as STATIC-KNOWN are expected to have
                well-known values, and therefore their values do not
                need to be communicated.

CHANGING   These fields are expected to vary randomly, either
                within a limited value set or range, or in some other
                manner. CHANGING fields are usually handled in more
                sophisticated ways based on a more detailed
                classification of their expected change patterns.

Finally, the last step is to choose the encoding method(s) that will
be applied onto different fields based on classification. The
encoding methods, in combination with the identified field behavior,
provide the input to the design of the compressed header formats.
The analysis of the probability distribution of the identified change
patterns then provides the means to optimize the packet formats,

where the most frequently occurring change patterns for a field
should be encoded within the most efficient format(s).

However, compression efficiency has to be traded against two other
properties: the robustness of the encoding to losses and errors
between the compressor and the decompressor, and the ability to
detect and cope with errors in the decompression process.

4.2. Compression Efficiency, Robustness, and Transparency

The performance of a header compression protocol can be described
with three parameters: its compression efficiency, its robustness,
and its compression transparency.

Compression efficiency

   The compression efficiency is determined by how much the average
   header size is reduced by applying the compression protocol.

Robustness

   A robust protocol tolerates packet losses, residual bit errors,
   and out-of-order delivery on the link over which header
   compression takes place, without losing additional packets or
   introducing additional errors in decompressed headers.

Compression transparency

   The compression transparency is a measure of the extent to which
   the scheme maintains the semantics of the original headers. If
   all decompressed headers are bitwise identical to the
   corresponding original headers, the scheme is transparent.

4.3. Developing the ROHC Protocol

The challenge in developing a header compression protocol is to
conciliate compression efficiency and robustness while maintaining
transparency, as increasing robustness will always come at the
expense of a lower compression efficiency, and vice-versa. The
scheme should also be flexible enough in its design to minimize the
impacts from the varying round-trip times and loss patterns of links
where header compression will be used.

To achieve this, the header compression scheme must provide
facilities for the decompressor to verify decompression and detect
potential context damage, as well as context recovery mechanisms such
as feedback. Header compression schemes prior to the ones developed

by the Robust Header Compression (ROHC) WG were not designed with the
above high-level objectives in mind.

The ROHC WG has developed header compression solutions to meet the
needs of present and future link technologies. While special
attention has been put towards meeting the more stringent
requirements stemming from the characteristics of wireless links, the
results are equally applicable to many other link technologies.

RFC 3095 [3], "RObust Header Compression (ROHC): Framework and four
profiles: RTP, UDP, ESP, and uncompressed", was published in 2001, as
the first output of the ROHC WG. ROHC is a general and extendable
framework for header compression, on top of which profiles can be
defined for compression of different protocols headers. RFC 3095
introduced a number of new compression techniques, and was successful
at living up to the requirements placed on it, as described in [18].

Interoperability testing of RFC 3095 confirms the capabilities of
ROHC to meet its purposes, but feedback from implementers has also
indicated that the protocol specification is complex and sometimes
obscure. Most importantly, a clear distinction between framework and
profiles is not obvious in [3], which also makes development of
additional profiles troublesome. This document therefore aims at
explicitly specifying the ROHC framework, while a companion document
[8] specifies revised versions of the compression profiles of RFC
3095.

4.4. Operational Characteristics of the ROHC Channel

Robust header compression can be used over many type of link
technologies. The ROHC framework provides flexibility for profiles
to address a wide range of applications, and this section lists some
of the operational characteristics of the ROHC channel (see also
[5]).

Multiplexing over a single logical channel

   The ROHC channel provides a mechanism to identify a context within
   the general ROHC packet format. The CID makes it possible for a
   logical channel that supports ROHC to transport multiple header-
   compressed flows, while still making it possible for a channel to
   be dedicated to one single packet flow without any CID overhead.
   More specifically, ROHC uses a distinct context identifier space
   per logical channel, and the context identifier can be omitted for
   one of the flows over the ROHC channel when configured to use a
   small CID space.

Establishment of channel parameters

   A link layer defining support for the ROHC channel must provide
   the means to establish header compression channel parameters (see
   Section 5.1). This can be achieved through a negotiation
   mechanism, static provisioning, or some out-of-band signaling.

Packet type identification

   The ROHC channel defines a packet type identifier space, and puts
   restrictions with respect to the use of a number of identifiers
   that are common for all ROHC profiles. Identifiers that have no
   restrictions, i.e., identifiers that are not defined by this
   document, are available to each profile. The identifier is part
   of each compressed header, and this makes it possible for the link
   that supports the ROHC channel to allocate one single link layer
   payload type for ROHC.

Out-of-order delivery between compression endpoints

   Each profile defines its own level of robustness, including
   tolerance to reordering of packets before but especially between
   compression endpoints, if any.

   For profiles specified in [3], the channel between the compressor
   and decompressor is required to maintain in-order delivery of the
   packets, i.e., the definition of these profiles assumes that the
   decompressor always receives packets in the same order as the
   compressor sent them. The impacts of reordering on the
   performance of these profiles is described in [7]. However,
   reordering before the compression point is handled, i.e., these
   profiles make no assumption that the compressor will receive
   packets in-order.

   For the ROHCv2 profiles specified in [8], their definitions assume
   that the decompressor can receive packets out-of-order, i.e., not
   in the same order that the compressor sent them. Reordering
   before the compression point is also dealt with.

Duplication of packets

   The link supporting the ROHC channel is required to not duplicate
   packets (however, duplication of packets can occur before they
   reach the compressor, i.e., there is no assumption that the
   compressor will receive only one copy of each packet).

Framing

   The link layer must provide framing that makes it possible to
   distinguish frame boundaries and individual frames.

Error detection/protection

   ROHC profiles should be designed to cope with residual errors in
   the headers delivered to the decompressor. CRCs are used to
   detect decompression failures and to prevent or reduce damage
   propagation. However, it is recommended that lower layers deploy
   error detection for ROHC headers and that ROHC headers with high
   residual error rates not be delivered.

4.5. Compression and Master Sequence Number (MSN)

   Compression of header fields is based on the establishment of a
   function to a sequence number, called the master sequence number
   (MSN). This function describes the change pattern of the field with
   respect to a change in the MSN.

   Change patterns include, for example, fields that increase
   monotonically or by a small value, fields that seldom change,and
   fields that remain unchanging for the entire lifetime of the packet
   flow, in which case the function to the MSN is equivalent to a
   constant value.

   The compressor first establishes functions for each of the header
   fields, and then reliably communicates the MSN. When the change
   pattern of the field does not match the established function, i.e.,
   the existing function gives a result that is different from the field
   in the header being compressed, additional information can be sent to
   update the parameters of that function.

   The MSN is defined per profile. It can be either derived directly
   from one of the fields of the protocol being compressed (e.g., the
   RTP SN [8]), or it can be created and maintained by the compressor
   (e.g., the MSN for compression of UDP in profile 0x0102 [8] or the
   MSN in ROHC-TCP [9]).

4.6. Static and Dynamic Parts of a Context

   A compression context can be conceptually divided into two different
   parts, the static context and the dynamic context, each based on the
   properties of the fields that are being compressed.

The static part includes the information necessary to compress and
decompress the fields whose change behavior is classified as STATIC,
STATIC-KNOWN, or STATIC-DEF (as described in Section 4.1 above).

The dynamic part includes the state maintained for all the other
fields, i.e., those that are classified as CHANGING.

5. The ROHC Framework (Normative)

This section normatively defines the parts common to all ROHC
profiles, i.e., the framework. The framework specifies the
requirements and functionality of the ROHC channel, including how to
handle multiple compressed packet flows over the same channel.

Finally, this section specifies encoding methods used in the packet
formats that are common to all profiles. These encoding methods may
be reused within profile specifications for encoding fields in
profile-specific parts of a packet format, without requiring their
redefinition.

5.1. The ROHC Channel

5.1.1. Contexts and Context Identifiers

Associated with each compressed flow is a context. The context is
the state that the compressor and the decompressor maintain in order
to correctly compress or decompress the headers of the packet in the
flow. Each context is identified using a CID.

A context is considered to be a new context when the CID is
associated with a profile for the first time since the creation of
the ROHC channel, or when the CID gets associated from the reception
of an IR (this does not apply to the IR-DYN) with a different profile
than the profile in the context.

Context information is conceptually kept in a table. The context
table is indexed using the CID, which is sent along with compressed
headers and feedback information.

The CID space can be either small, which means that CIDs can take the
values 0 through 15, or large, which means that CIDs take values
between 0 and 2^14 - 1 = 16383. Whether the CID space is large or
small MUST be established, possibly by negotiation, before any
compressed packet may be sent over the ROHC channel.

The CID space is distinct for each channel, i.e., CID 3 over channel
A and CID 3 over channel B do not refer to the same context, even if
the endpoints of A and B are the same nodes. In particular, CIDs for

any pair of ROHC channels are not related (two associated ROHC
channels serving as feedback channels for one another do not even
need to have CID spaces of the same size).

5.1.2. Per-Channel Parameters

The ROHC channel is based on a number of parameters that form part of
the established channel state and the per-context state. The state
of the ROHC channel MUST be established before the first ROHC packet
may be sent, which may be achieved using negotiation protocols
provided by the link layer (see also [4], which describes an option
for negotiation of ROHC parameters for PPP). This section describes
some of this channel state information in an abstract way:

LARGE_CIDS: Boolean; if false, the small CID representation (0 octets
   or 1 prefix octet, covering CID 0 to 15) is used; if true, the
   large CID representation (1 or 2 embedded CID octets covering CID
   0 to 16383) is used. See also 5.1.1 and 5.2.1.3.

MAX_CID: Non-negative integer; highest CID number to be used by the
   compressor (note that this parameter is not coupled to, but in
   effect further constrained by, LARGE_CIDS). This value represents
   an agreement by the decompressor that it can provide sufficient
   memory resources to host at least MAX_CID+1 contexts; the
   decompressor MUST maintain established contexts within this space
   until either the CID gets re-used by the establishment of a new
   context, or until the channel is taken down.

PROFILES: Set of non-negative integers, where each integer indicates
   a profile supported by both the compressor and the decompressor.
   A 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. The ROHC compressed header format
   identifies the profile used with only the 8 LSB bits; this means
   that if multiple variants of the same profile are available for a
   ROHC channel, the PROFILES set after negotiation MUST NOT include
   more than one variant of the same profile. The compressor MUST
   NOT compress using a profile that is not in PROFILES.

FEEDBACK_FOR: Optional reference to a ROHC channel in the opposite
   direction between the same compression endpoints. If provided,
   this parameter indicates to which other ROHC channel any feedback
   sent on this ROHC channel refers (see [5]).

MRRU: Non-negative integer. Maximum Reconstructed Reception Unit.
   This is the size of the largest reconstructed unit in octets that
   the decompressor is expected to reassemble from segments (see
   Section 5.2.5). This size includes the segmentation CRC. If MRRU

is negotiated to be 0, segmentation MUST NOT be used on the
channel, and received segments MUST be discarded by the
decompressor.

5.1.3. Persistence of Decompressor Contexts

As part of the negotiated channel parameters, the compressor and
decompressor have through the MAX_CID parameter agreed on the highest
context identification (CID) number to be used. By agreeing on the
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 SHOULD be kept by the decompressor
until either the CID gets re-used, or the channel is taken down or
re-negotiated.

5.2. ROHC Packets and Packet Types

This section uses the following convention in the diagrams when
representing various ROHC packet types, formats, and fields:

    - colons ":" indicate that the part is optional
    - slashes "/" indicate variable length

The ROHC packet type indication scheme has been designed to provide
optional padding, a feedback packet type, an optional Add-CID octet
(which includes 4 bits of CID), and a simple segmentation and
reassembly mechanism.

The following packet types are reserved at the ROHC framework level:

    11100000 : Padding
    1110nnnn : Add-CID octet (nnnn=CID with values 0x1 through 0xF)
    11110   : Feedback
    11111000 : IR-DYN packet
    1111110 : IR packet
    1111111 : Segment

Other packet types can be defined and used by individual profiles:

    0   : available (not reserved by ROHC framework)
    10   : available (not reserved by ROHC framework)
    110   : available (not reserved by ROHC framework)
    1111101 : available (not reserved by ROHC framework)
    11111001 : available (not reserved by ROHC framework)

5.2.1. General Format of ROHC Packets

   A ROHC packet has the following general format:

```
    --- --- --- --- --- --- --- ---
   :              Padding          :
    --- --- --- --- --- --- --- ---
   :              Feedback         :
    --- --- --- --- --- --- --- ---
   :               Header          :
    --- --- --- --- --- --- --- ---
   :               Payload         :
    --- --- --- --- --- --- --- ---
```

   Padding: Any number (zero or more) of padding octets, where the
      format of a padding octet is as defined in Section 5.2.1.1.

   Feedback: Any number (zero or more) of feedback elements, where the
      format of a feedback element is as defined in Section 5.2.4.1.

   Header: Either a profile-specific CO header (see Section 5.2.1.3), an
      IR or IR-DYN header (see Section 5.2.2), or a ROHC Segment (see
      Section 5.2.5). There can be at most one Header in a ROHC packet,
      but it may also be omitted (if the packet contains Feedback only).

   Payload: Corresponds to zero or more octets of payload from the
      uncompressed packet, starting with the first octet in the
      uncompressed packet after the last header compressible by the
      current profile.

   At least one of Feedback or Header MUST be present.

5.2.1.1. Format of the Padding Octet

   Padding octet:

```
     0   1   2   3   4   5   6   7
   +---+---+---+---+---+---+---+---+
   | 1   1   1   0   0   0   0   0 |
   +---+---+---+---+---+---+---+---+
```

   Note: The Padding octet MUST NOT be interpreted as an Add-CID octet
   for CID 0.

5.2.1.2. Format of the Add-CID Octet

   Add-CID octet:

      0   1   2   3   4   5   6   7
    +---+---+---+---+---+---+---+---+
    | 1   1   1   0 | CID |
    +---+---+---+---+---+---+---+---+

   CID: 0x1 through 0xF indicates CIDs 1 through 15.

   Note: The Padding octet looks like an Add-CID octet for CID 0.

5.2.1.3. General Format of Header

   All ROHC packet types have the following general Header format:

      0   x-1 x   7
     --- --- --- --- --- --- --- ---
     :   Add-CID octet   : if CID 1-15 and small CIDs
    +--- --- --- --- ---+--- --- ---+
    | type indication | body | 1 octet (8-x bits of body)
    +--- --- --- --- ---+--- --- ---+
    :   :
    /   0, 1, or 2 octets of CID   / 1 or 2 octets if large CIDs
    :   :
    +---+---+---+---+---+---+---+---+
    /   body   / variable length
    +---+---+---+---+---+---+---+---+

   type indication: ROHC packet type.

   body: Interpreted according to the packet type indication and CID
      information, as defined by individual profiles.

   Thus, the header either starts with a packet type indication or has a
   packet type indication immediately following an Add-CID octet.

   When the ROHC channel is configured with a small CID space:

      o If an Add-CID immediately precedes the packet type indication,
         the packet has the CID of the Add-CID; otherwise, it has CID 0.

      o A small CID with the value 0 is represented using zero bits;
         therefore, a flow associated with CID 0 has no CID overhead in
         the compressed header. In such case, Header starts with a
         packet type indication.

   o A small CID with a value from 1 to 15 is represented using the
      Add-CID octet as described above. The Header starts with the
      Add-CID octet, followed by a packet type indication.

   o There is no large CID in the Header.

When the ROHC channel is configured with a large CID space:

   o The large CID is always present and is represented using the
      encoding scheme of Section 5.3.2, limited to two octets. In
      this case, the Header starts with a packet type indication.

5.2.2. Initialization and Refresh (IR) Packet Types

IR packet types contain a profile identifier, which determines how
the rest of the header is to be interpreted. They also associate a
profile with a context. The stored profile parameter further
determines the syntax and semantics of the packet type identifiers
and packet types used with a specific context.

The IR and IR-DYN packets always update the context for all context-
updating fields carried in the header. They never clear the context,
except when initializing a new context (see Section 5.1.1), or unless
the profile indicated in the Profile field specifies otherwise.

5.2.2.1. ROHC IR Packet Type

   The IR header associates a CID with a profile, and typically also
   initializes the context. It can typically also refresh all (or parts
   of) the context. For IR, Header has the following general format:

```
     0   1   2   3   4   5   6   7
   --- --- --- --- --- --- --- ---
  :           Add-CID octet         :  if CID 1-15 and small CID
  +---+---+---+---+---+---+---+---+
  | 1   1   1   1   1   1   0 | x |  IR type octet
  +---+---+---+---+---+---+---+---+
  :                               :
  /        0-2 octets of CID       /  1 or 2 octets if large CIDs
  :                               :
  +---+---+---+---+---+---+---+---+
  |            Profile             |  1 octet
  +---+---+---+---+---+---+---+---+
  |              CRC               |  1 octet
  +---+---+---+---+---+---+---+---+
  |                               |
  / profile specific information   /  variable length
  |                               |
  +---+---+---+---+---+---+---+---+
```
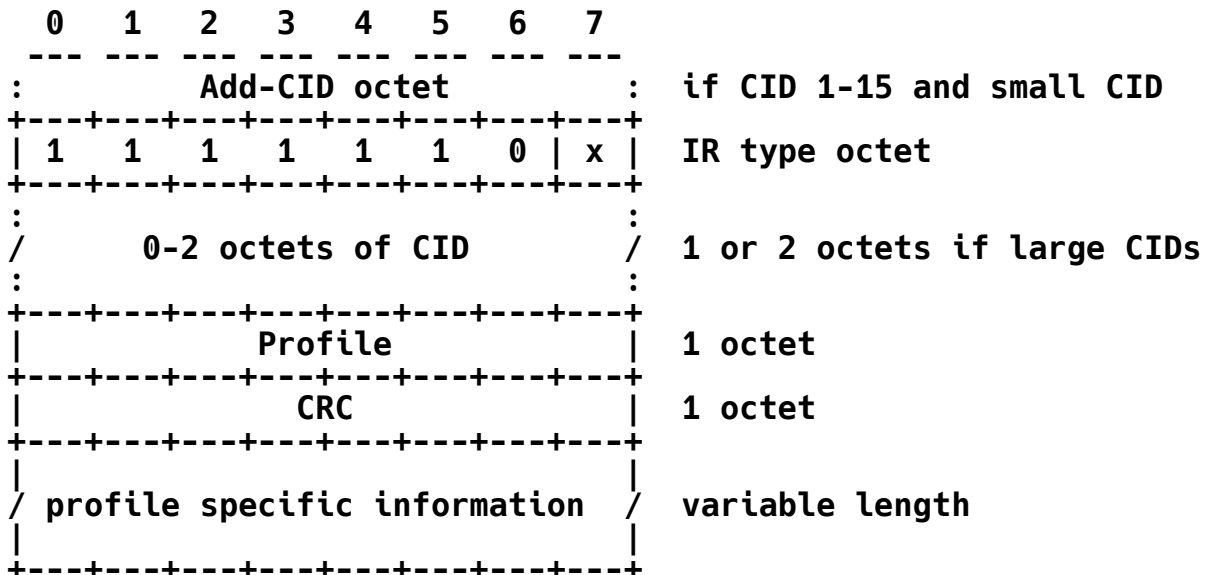
   x: Profile specific information. Interpreted according to the
      profile indicated in the Profile field of the IR header.

   Profile: The profile associated with the CID. In the IR header, the
      profile identifier is abbreviated to the 8 least significant bits
      (see Section 5.1.2).

   CRC: 8-bit CRC (see Section 5.3.1.1).

   Profile specific information: The content of this part of the IR
      header is defined by the individual profiles. It is interpreted
      according to the profile indicated in the Profile field.

5.2.2.2. ROHC IR-DYN Packet Type

   In contrast to the IR header, the IR-DYN header can never initialize
   a non-initialized context. However, it can redefine what profile is
   associated with a context, if the profile indicated in the IR-DYN
   header allows this. Thus, this packet type is also reserved at the
   framework level. The IR-DYN header typically also initializes or
   refreshes parts of a context. For IR-DYN, Header has the following
   general format:

```
       0   1   2   3   4   5   6   7
     --- --- --- --- --- --- --- ---
   :          Add-CID octet         :  if CID 1-15 and small CID
   +---+---+---+---+---+---+---+---+
   | 1   1   1   1   1   0   0   0 |  IR-DYN type octet
   +---+---+---+---+---+---+---+---+
   :                               :
   /      0-2 octets of CID        /  1 or 2 octets if large CIDs
   :                               :
   +---+---+---+---+---+---+---+---+
   |            Profile            |  1 octet
   +---+---+---+---+---+---+---+---+
   |              CRC              |  1 octet
   +---+---+---+---+---+---+---+---+
   |                               |
   / profile specific information  /  variable length
   |                               |
   +---+---+---+---+---+---+---+---+
```

   Profile: The profile associated with the CID. This is abbreviated in
      the same way as in IR packets.

   CRC: 8-bit CRC (see Section 5.3.1.1).

   Profile specific information: The content of this part of the IR-DYN
      header is defined by the individual profiles. It is interpreted
      according to the profile indicated in the Profile field.

5.2.3. ROHC Initial Decompressor Processing

   Initially, all contexts are in no context state. Thus, all packets
   referencing a non-initialized context, except packets that have
   enough information on the static fields, cannot be decompressed by
   the decompressor.

   When the decompressor receives a packet of type IR, the profile
   indicated in the IR packet determines how it is to be processed.

      o If the 8-bit CRC fails to verify the integrity of the Header,
         the packet MUST NOT be decompressed and delivered to upper
         layers. If a profile is indicated in the context, the logic of
         that profile determines what, if any, feedback is to be sent.
         If no profile is noted in the context, the logic used to
         determine what, if any, feedback to send is up to the
         implementation. However, it may be suitable to take no further
         actions, as any part of the IR header covered by the CRC may
         have caused the failure.

When the decompressor receives a packet of type IR-DYN, the profile
indicated in the IR-DYN packet determines how it is to be processed.

   o If the 8-bit CRC fails to verify the integrity of the header,
      the packet MUST NOT be decompressed and delivered to upper
      layers. If a profile is indicated in the context, the logic of
      that profile determines what, if any, feedback is to be sent.
      If no profile is noted in the context, the logic used to
      determine what, if any, feedback to send is up to the
      implementation. However, it may be suitable to take no further
      actions, as any part of the IR-DYN header covered by the CRC
      may have caused the failure.

   o If the context has not already been initialized, the packet
      MUST NOT be decompressed and delivered to upper layers. The
      logic of the profile indicated in the IR-DYN header (if
      verified by the 8-bit CRC), determines what, if any, feedback
      is to be sent.

If a parsing error occurs for any packet type, the decompressor MUST
discard the packet without further processing. For example, a CID
field is present in the compressed header when the large CID space is
used for the ROHC channel, and the field is coded using the self-
describing variable-length encoding of Section 5.3.2; if the field
starts with 110 or 111, this would generate a parsing error for the
decompressor because this field must not be encoded with a size
larger than 2 octets.

It is RECOMMENDED that profiles disallow the decompressor to make a
decompression attempt for packets carrying only a 3-bit CRC after it
has invalidated some or all of the entire dynamic context, until a
packet that contains sufficient information on the dynamic fields is
received, decompressed, and successfully verified by a 7- or 8-bit
CRC.

5.2.4. ROHC Feedback

Feedback carries information from the decompressor to compressor.
Feedback can be sent over a ROHC channel that operates in the same
direction as the feedback.

The general ROHC packet format allows transport of feedback using
interspersion or piggybacking (see [5]), or a combination of both,
over a ROHC channel. This is facilitated by the following
properties:

   Reserved packet type:

      A feedback packet type is reserved at the framework level. The
      packet type can carry variable-length feedback information.

   CID information:

      The feedback information sent on a particular channel is passed
      to, and interpreted by, the compressor associated with feedback on
      that channel. Thus, each feedback element contains CID
      information from the channel for which the feedback is sent. The
      ROHC feedback scheme thus requires that a channel carries feedback
      to at most one compressor. How a compressor is associated with
      the feedback for a particular channel is outside the scope of this
      specification. See also [5].

   Length information:

      The length of a feedback element can be determined by examining
      the first few octets of the feedback. This enables piggybacking
      of feedback, and also the concatenation of more than one feedback
      element in a packet. The length information thus decouples the
      decompressor from the associated same-side compressor, as the
      decompressor can extract the feedback information from the
      compressed header without parsing its content and hand over the
      extracted information.

   The association between compressor-decompressor pairs operating in
   opposite directions, for the purpose of exchanging piggyback and/or
   interspersed feedback, SHOULD be maintained for the lifetime of the
   ROHC channel. Otherwise, it is RECOMMENDED that the compressor be
   notified if the feedback channel is no longer available: the
   compressor SHOULD then restart compression by creating a new context
   for each packet flow, and SHOULD use a CID value that was not
   previously associated with the profile used to compress the flow.

5.2.4.1. ROHC Feedback Format

   ROHC defines three different categories of feedback messages:
   acknowledgment (ACK), negative ACK (NACK), and NACK for the entire
   context (STATIC-NACK). Other types of information may be defined in
   profile-specific feedback information.

   ACK   : Acknowledges successful decompression of a packet.
                 Indicates that the decompressor considers its context
                 to be valid.

NACK   : Indicates that the decompressor considers some or all
              of the dynamic part of its context invalid.

STATIC-NACK : Indicates that the decompressor considers its entire
              static context invalid, or that it has not been
              established.

Feedback sent on a ROHC channel consists of one or more concatenated
feedback elements, where each feedback element has the following
format:

```
     0   1   2   3   4   5   6   7
   +---+---+---+---+---+---+---+---+
   | 1   1   1   1   0 |  Code     |   feedback type
   +---+---+---+---+---+---+---+---+
   :            Size            :   if Code = 0
   +---+---+---+---+---+---+---+---+
   :        Add-CID octet       :   if for small CIDs and (CID != 0)
   +---+---+---+---+---+---+---+---+
   :                            :
   /  large CID (5.3.2 encoding) /   1-2 octets if for large CIDs
   :                            :
   +---+---+---+---+---+---+---+---+
   /        FEEDBACK data        /   variable length
   +---+---+---+---+---+---+---+---+
```

Code: 0 indicates that a Size octet is present.
      1-7 indicates the size of the feedback data field, in octets.

Size: Indicates the size of the feedback data field, in octets.

FEEDBACK data: FEEDBACK-1 or FEEDBACK-2 (see below).

CID information in a feedback element indicates the context for which
feedback is sent. The LARGE_CIDS parameter that controls whether a
large CID is present is taken from the channel state of the receiving
compressor's channel, not from the state of the channel carrying the
feedback.

The large CID field, if present, is encoded according to Section
5.3.2, and it MUST NOT be encoded using more than 2 octets.

The FEEDBACK data field can have either of the following two formats:

FEEDBACK-1:

```
  0   1   2   3   4   5   6   7
+---+---+---+---+---+---+---+---+
| profile specific information  |  1 octet
+---+---+---+---+---+---+---+---+
```

FEEDBACK-2:

```
  0   1   2   3   4   5   6   7
+---+---+---+---+---+---+---+---+
|Acktype|                       |
+---+---+   profile specific    /  at least 2 octets
/           information         |
+---+---+---+---+---+---+---+---+
```

```
Acktype:  0 = ACK
          1 = NACK
          2 = STATIC-NACK
          3 is reserved (MUST NOT be used.  Otherwise unparseable.)
```

5.2.5. ROHC Segmentation

   ROHC defines a simple segmentation protocol. The compressor may
   perform segmentation, e.g., to accommodate packets that are larger
   than a specific size configured for the channel.

5.2.5.1. Segmentation Usage Considerations

   The ROHC segmentation protocol is not particularly efficient. It is
   not intended to replace link layer segmentation functions; these
   SHOULD be used whenever available and efficient for the task at hand.
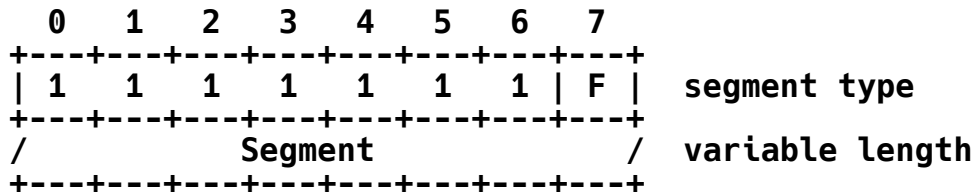
   The ROHC segmentation protocol has been designed with an assumption
   of in-order delivery of packets between the compressor and the
   decompressor, using only a CRC for error detection, and no sequence
   numbers. If in-order delivery cannot be guaranteed, ROHC
   segmentation MUST NOT be used.

   The segmentation protocol also assumes that all segments of a ROHC
   packet corresponding to one context are received without interference
   from other ROHC packets over the channel, including any ROHC packet
   corresponding to a different context. Based on this assumption,
   segments do not carry CID information, and therefore cannot be
   associated with a specific context until all segments have been
   received and the whole unit has been reconstructed.

5.2.5.2. Segmentation Protocol

   ROHC segmentation is applied to the combination of the Header and the
   Payload fields of the ROHC packet, as defined in Section 5.2.1.

   Segment format:

      0   1   2   3   4   5   6   7
    +---+---+---+---+---+---+---+---+
    | 1   1   1   1   1   1   1 | F | segment type
    +---+---+---+---+---+---+---+---+
    /   Segment   /   variable length
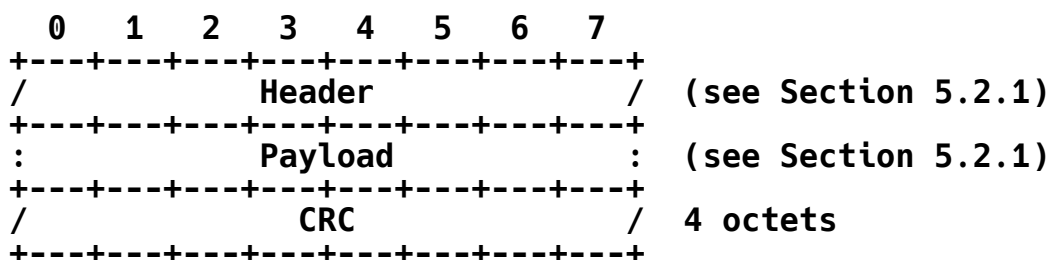   +---+---+---+---+---+---+---+---+

   F: Final bit. If set, it indicates that this is the last segment of
      a reconstructed unit.

   Padding and/or Feedback may precede the segment type octet. There is
   no per-segment CID, but CID information is of course part of the
   reconstructed unit. The reconstructed unit MUST NOT contain padding,
   segments, or feedback.

   When a final segment is received, the decompressor reassembles the
   segment carried in this packet and any non-final segments that
   immediately preceded it into a single reconstructed unit, in the
   order they were received. All segments for one reconstructed unit
   have to be received consecutively and in the correct order by the
   decompressor. If a non-segment ROHC packet directly follows a non-
   final segment, the reassembly of the current reconstructed unit is
   aborted and the decompressor MUST discard the non-final segments so
   far received on this channel.

   Reconstructed unit:

      0   1   2   3   4   5   6   7
    +---+---+---+---+---+---+---+---+
    /   Header   /   (see Section 5.2.1)
    +---+---+---+---+---+---+---+---+
    :   Payload   :   (see Section 5.2.1)
    +---+---+---+---+---+---+---+---+
    /   CRC   /   4 octets
    +---+---+---+---+---+---+---+---+

   CRC: 32-bit CRC computed using the polynomial of Section 5.3.1.4.

   If the reconstructed unit is 4 octets or less, or if the CRC fails,
   or if it is larger than the channel parameter MRRU (see Section

5.1.2), the reconstructed unit MUST be discarded by the decompressor. If the CRC succeeds, the reconstructed unit can be further processed.

5.3. General Encoding Methods

5.3.1. Header Compression CRCs, Coverage and Polynomials

This section describes how to calculate the CRCs used by ROHC. For all CRCs, the algorithm used to calculate the CRC is the same as the one used in [2], defined in Appendix A of this document, with the polynomials specified in subsequent sections.

5.3.1.1. 8-bit CRCs in IR and IR-DYN Headers

The coverage for the 8-bit CRC in the IR and IR-DYN headers is profile-dependent, but it MUST cover at least the initial part of the header ending with the Profile field, including the CID or an Add-CID octet. Feedback and padding are not part of Header (Section 5.2.1) and are thus not included in the CRC calculation. As a rule of thumb for profile specifications, any other information that initializes the decompressor context SHOULD also be covered by a CRC.

More specifically, the 8-bit CRC does not cover only and entirely the original uncompressed header; therefore, it does not provide the means for the decompressor to verify a decompression attempt, or the means to verify the correctness of the entire decompressor context. However, when successful, it does provide enough robustness for the decompressor to update its context with the information carried within the IR or the IR-DYN header.

The CRC polynomial for the 8-bit CRC is:

   $C(x) = 1 + x + x^2 + x^8$

When computing the CRC, the CRC field in the header is set to zero, and the initial content of the CRC register is set to all 1's.

5.3.1.2. 3-bit CRC in Compressed Headers

The 3-bit CRC in compressed headers is calculated over all octets of the entire original header, before compression, in the following manner.

The initial content of the CRC register is set to all 1's.

The polynomial for the 3-bit CRC is:

   $C(x) = 1 + x + x^3$

The purpose of the 3-bit CRC is to provide the means for the
decompressor to verify the outcome of a decompression attempt for
small compressed headers, and to detect context damage based on
aggregated probability over a number of decompression attempts. It
is however too weak to provide enough success guarantees from the
decompression of one single header. Therefore, compressed headers
carrying a 3-bit CRC are normally not suitable to perform context
repairs at the decompressor; hence, profiles should refrain from
allowing decompression of such a header when some or the entire
decompressor context is assumed invalid.

5.3.1.3. 7-bit CRC in Compressed Headers

The 7-bit CRC in compressed headers is calculated over all octets of
the entire original header, before compression, in the following
manner.

The initial content of the CRC register is set to all 1's.

The polynomial for the 7-bit CRC is:

$$C(x) = 1 + x + x^2 + x^3 + x^6 + x^7$$

The purpose of the 7-bit CRC is to provide the means for the
decompressor to verify the outcome of a decompression attempt for a
larger compressed header, and to provide enough protection to
validate a context repair at the decompressor. The 7-bit CRC is
strong enough to assume a repair to be successful from the
decompression of one single header; hence, profiles may allow
decompression of a header carrying a 7-bit CRC when some of the
decompressor context is assumed invalid.

5.3.1.4. 32-bit Segmentation CRC

The 32-bit CRC is used by the segmentation scheme to verify the
reconstructed unit, and it is thus calculated over the segmented
unit, i.e., over the Header and the Payload fields of the ROHC
packet.

The initial content of the CRC register is set to all 1's.

The polynomial for the 32-bit CRC is:

$$C(x) = x^0 + x^1 + x^2 + x^4 + x^5 + x^7 + x^8 + x^{10} +$$
$$x^{11} + x^{12} + x^{16} + x^{22} + x^{23} + x^{26} + x^{32}.$$

The purpose of the 32-bit CRC is to verify the reconstructed unit.

5.3.2. Self-Describing Variable-Length Values

The values of many fields and compression parameters can vary widely.
To optimize the transfer of such values, a variable number of octets
are used to encode them. The first few bits of the first octet
determine the number of octets used:

First bit is 0: 1 octet.
        7 bits transferred.
        Up to 127 decimal.
        Encoded octets in hexadecimal: 00 to 7F

First bits are 10: 2 octets.
        14 bits transferred.
        Up to 16 383 decimal.
        Encoded octets in hexadecimal: 80 00 to BF FF

First bits are 110: 3 octets.
        21 bits transferred.
        Up to 2 097 151 decimal.
        Encoded octets in hexadecimal: C0 00 00 to DF FF FF

First bits are 111: 4 octets.
        29 bits transferred.
        Up to 536 870 911 decimal.
        Encoded octets in hexadecimal: E0 00 00 00 to FF FF FF FF

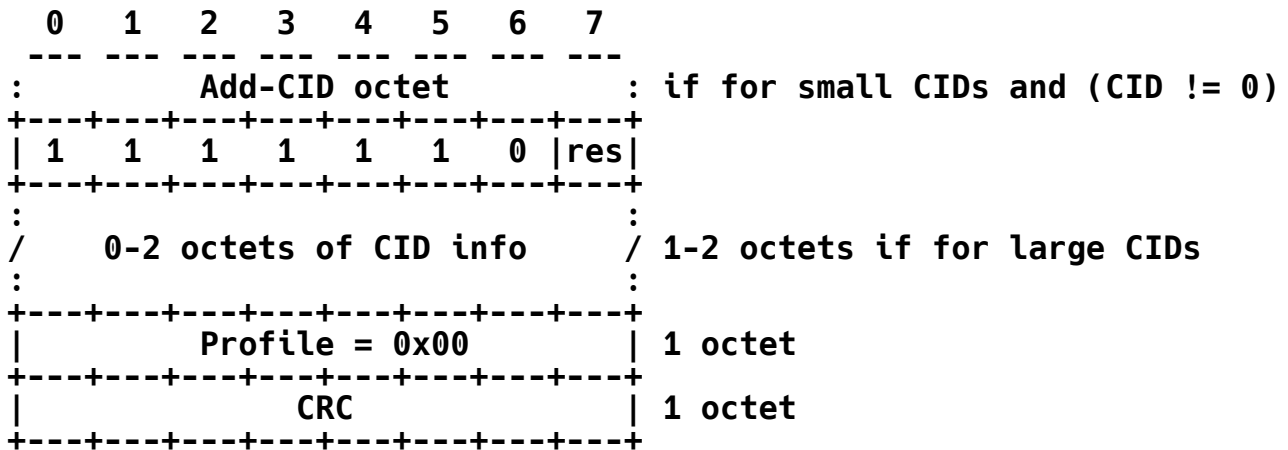## 5.4. ROHC UNCOMPRESSED -- No Compression (Profile 0x0000)

This section describes the uncompressed ROHC profile. The profile
identifier for this profile is 0x0000.

Profile 0x0000 provides a way to send IP packets without compressing
them. This can be used for any packet for which a compression
profile is not available in the set of profiles supported by the ROHC
channel, or for which compression is not desirable for some reason.

After initialization, the only overhead for sending packets using
Profile 0x0000 is the size of the CID. When uncompressed packets are
frequent, Profile 0x0000 should be associated with a CID the size of
zero or one octet. Profile 0x0000 SHOULD be associated with at most
one CID.

5.4.1. IR Packet

   The initialization and refresh packet (IR packet) for Profile 0x0000
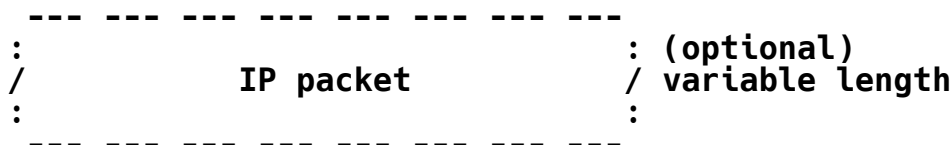   has the following Header format:

```
     0   1   2   3   4   5   6   7
   --- --- --- --- --- --- --- ---
  :           Add-CID octet         : if for small CIDs and (CID != 0)
  +---+---+---+---+---+---+---+---+
  | 1   1   1   1   1   1   0 |res|
  +---+---+---+---+---+---+---+---+
  :                               :
  /     0-2 octets of CID info    / 1-2 octets if for large CIDs
  :                               :
  +---+---+---+---+---+---+---+---+
  |         Profile = 0x00        | 1 octet
  +---+---+---+---+---+---+---+---+
  |              CRC              | 1 octet
  +---+---+---+---+---+---+---+---+
```

   res: MUST be set to zero; otherwise, the decompressor MUST discard
        the packet.

   Profile: 0x00

   CRC: 8-bit CRC, computed using the polynomial of Section 5.3.1.1.
        The CRC covers the first octet of the IR Header through the
        Profile octet of the IR Header, i.e., it does not cover the CRC
        itself. Neither does it cover any preceding Padding or
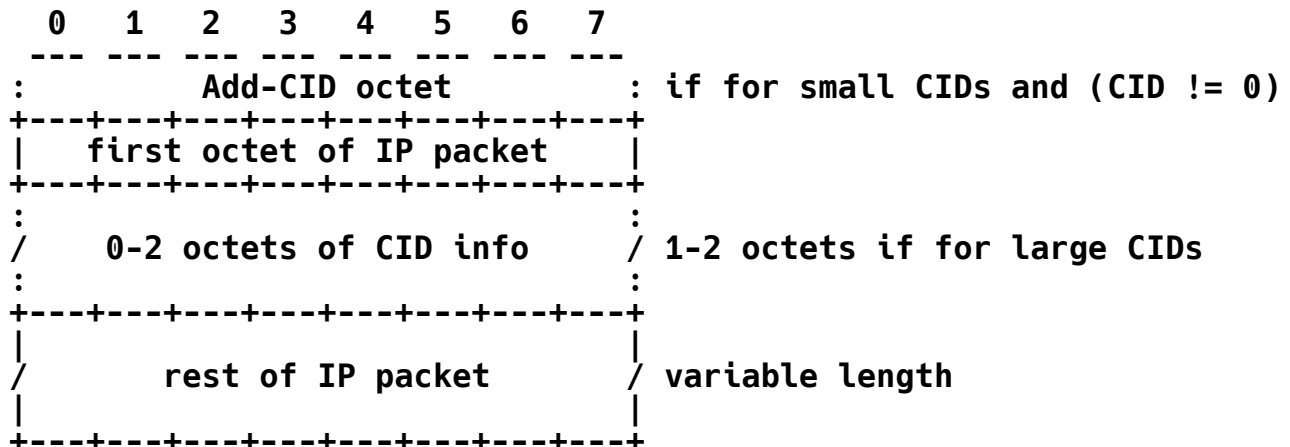   Feedback, nor the Payload.

   For the IR packet, Payload has the following format:

```
   --- --- --- --- --- --- --- ---
  :                               : (optional)
  /           IP packet           / variable length
  :                               :
   --- --- --- --- --- --- --- ---
```

   IP packet: An uncompressed IP packet may be included in the IR
      packet. The decompressor determines if the IP packet is present
      by considering the length of the IR packet.

5.4.2. Normal Packet

A Normal packet is a normal IP packet plus CID information. For the
Normal Packet, the following format corresponds to the Header and
Payload (as defined in Section 5.2.1):

```
     0   1   2   3   4   5   6   7
    --- --- --- --- --- --- --- ---
   :           Add-CID octet        : if for small CIDs and (CID != 0)
   +---+---+---+---+---+---+---+---+
   |    first octet of IP packet   |
   +---+---+---+---+---+---+---+---+
   :                               :
   /     0-2 octets of CID info    / 1-2 octets if for large CIDs
   :                               :
   +---+---+---+---+---+---+---+---+
   |                               |
   /         rest of IP packet     / variable length
   |                               |
   +---+---+---+---+---+---+---+---+
```

Note that the first octet of the IP packet starts with the bit
pattern 0100 (IPv4) or 0110 (IPv6). This does not conflict with any
reserved packet types.

When the channel uses small CIDs, and profile 0x0000 is associated
with a CID > 0, an Add-CID octet precedes the IP packet. When the
channel uses large CIDs, the CID is placed so that it starts at the
second octet of the combined Header/Payload format above.

A Normal Packet may carry Padding and/or Feedback as any other ROHC
packet, preceding the combined Header/Payload.

5.4.3. Decompressor Operation

When an IR packet is received, the decompressor first validates its
header using the 8-bit CRC.

o If the header fails validation, the decompressor MUST NOT deliver
   the IP packet to upper layers.

o If the header is successfully validated, the decompressor

       1) initializes the context if it has no valid context for the
          given CID already associated to the specified profile,

       2) delivers the IP packet to upper layers if present,

      3) MAY send an ACK.

   When any other packet is received while the decompressor has no
   context, it is discarded without further action.

   When a Normal packet is received and the decompressor has a valid
   context, the IP packet is extracted and delivered to upper layers.

5.4.4. Feedback

   The only kind of feedback defined by Profile 0x0000 is ACK, using the
   FEEDBACK-1 format of Section 5.2.4.1, where the value of the profile-
   specific octet in the FEEDBACK-1 is 0 (zero). The FEEDBACK-2 format
   is thus not defined for Profile 0x0000.

6. Overview of a ROHC Profile (Informative)

   The ROHC protocol consists of a framework part and a profile part.
   The framework defines the mechanisms common to all profiles, while
   the profile defines the compression algorithm and profile specific
   packet formats.

   Section 5 specifies the details of the ROHC framework. This section
   provides an informative overview of the elements that make a profile
   specification. The normative specification of individual profiles is
   outside the scope of this document.

   A ROHC profile defines the elements that build up the compression
   protocol. A ROHC profile consists of:

   Packet formats:

   o Bits-on-the-wire

      The profile defines the layout of the bits for profile-specific
      packet types that it defines, and for the profile-specific parts
      of packet types common to all profiles (e.g., IR and IR-DYN).

   o Field encodings

      Bits and groups of bits from the packet format layout, referred to
      as Compressed fields, represents the result of an encoding method
      specific for that compressed field within a specific packet
      format. The profile defines these encoding methods.

o Updating properties

   The profile-specific packet formats may update the state of the
   decompressor, and may do so in different ways. The profile
   defines how individual profile-specific fields, or entire
   profile-specific packet types, update the decompressor context.

o Verification

   Packets that update the state of the decompressor are verified to
   prevent incorrect updates to the decompressor context. The
   profile defines the mechanisms used to verify the decompression of
   a packet.

Context management:

o Robustness logic

   Packets may be lost or reordered between the compressor and the
   decompressor. The profile defines mechanism to minimize the
   impacts of such events and prevent damage propagation.

o Repair mechanism

   Despite the robustness logic, impairment events may still lead to
   decompression failure(s), and even to context damage at the
   decompressor. The profile defines context repair mechanisms,
   including feedback logic if used.

7. Security Considerations

   Because encryption eliminates the redundancy that header compression
   schemes try to exploit, there is some inducement to forego encryption
   of headers in order to enable operation over low-bandwidth links.

   A malfunctioning or malicious header compressor could cause the
   header decompressor to reconstitute packets that do not match the
   original packets but still have valid headers and possibly also valid
   transport checksums. Such corruption may be detected with end-to-end
   authentication and integrity mechanisms, which will not be affected
   by the compression. Moreover, the ROHC header compression scheme
   uses an internal checksum for verification of reconstructed headers,
   which reduces the probability of producing decompressed headers not
   matching the original ones without this being noticed.

   Denial-of-service attacks are possible if an intruder can introduce,
   for example, bogus IR, IR-DYN, or FEEDBACK packets onto the link and
   thereby cause compression efficiency to be reduced. However, an

   intruder having the ability to inject arbitrary packets at the link
   layer in this manner raises additional security issues that dwarf
   those related to the use of header compression.

8. IANA Considerations

   An IANA registry for "RObust Header Compression (ROHC) Profile
   Identifiers" [21] was created by RFC 3095 [3]. The assignment
   policy, as outlined by RFC 3095, is the following:

   The ROHC profile identifier is a non-negative integer. In many
   negotiation protocols, it will be represented as a 16-bit value. Due
   to the way the profile identifier is abbreviated in ROHC packets, the
   8 least significant bits of the profile identifier have a special
   significance: Two profile identifiers with identical 8 LSBs should be
   assigned only if the higher-numbered one is intended to supersede the
   lower-numbered one. To highlight this relationship, profile
   identifiers should be given in hexadecimal (as in 0x1234, which would
   for example supersede 0x0A34).

   Following the policies outlined in [22], the IANA policy for
   assigning new values for the profile identifier shall be
   Specification Required: values and their meanings must be documented
   in an RFC or in some other permanent and readily available reference,
   in sufficient detail that interoperability between independent
   implementations is possible. In the 8 LSBs, the range 0 to 127 is
   reserved for IETF standard-track specifications; the range 128 to 254
   is available for other specifications that meet this requirement
   (such as Informational RFCs). The LSB value 255 is reserved for
   future extensibility of the present specification.

   The following profile identifiers have so far been allocated:

   Profile Identifier   Usage   Reference
   ------------------   ----------------------   ---------
   0x0000   ROHC uncompressed   RFC 4995
   0x0001   ROHC RTP   RFC 3095
   0x0002   ROHC UDP   RFC 3095
   0x0003   ROHC ESP   RFC 3095
   0x0004   ROHC IP   RFC 3843
   0x0005   ROHC LLA   RFC 3242
   0x0105   ROHC LLA with R-mode   RFC 3408
   0x0006   ROHC TCP   RFC 4996
   0x0007   ROHC RTP/UDP-Lite   RFC 4019
   0x0008   ROHC UDP-Lite   RFC 4019

   New profiles will need new identifiers to be assigned by the IANA,
   but this document does not require any additional IANA action.

9. Acknowledgments

10. References

10.1. Normative References

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

10.2. Informative References

[2] Simpson, W., "PPP in HDLC-like Framing", STD 51, RFC 1662, July 1994.

[3] 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.

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

[5] Jonsson, L-E., "RObust Header Compression (ROHC): Terminology and Channel Mapping Examples", RFC 3759, April 2004.

[6] Jonsson, L-E., Sandlund, K., Pelletier, G., and P. Kremer, "RObust Header Compression (ROHC): Corrections and Clarifications to RFC 3095", RFC 4815, February 2007.

[7] Pelletier, G., Jonsson, L-E., and K. Sandlund, "RObust Header Compression (ROHC): ROHC over Channels That Can Reorder Packets", RFC 4224, January 2006.

[8] Pelletier, G. and K. Sandlund, "RObust Header Compression Version 2 (ROHCv2): Profiles for RTP, UDP, IP, ESP, and UDP Lite", Work in Progress, September 2006.

[9]   Pelletier, G., Sandlund, K., Jonsson, L-E., and M. West, "RObust
      Header Compression (ROHC): A Profile for TCP/IP (ROHC-TCP)", RFC
      4996, July 2007.

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

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

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

[13] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson,
      "RTP: A Transport Protocol for Real-Time Applications", STD 64,
      RFC 3550, July 2003.

[14] Postel, J., "Transmission Control Protocol", STD 7, RFC 793,
      September 1981.

[15] Jacobson, V., "Compressing TCP/IP headers for low-speed serial
      links", RFC 1144, February 1990.

[16] Degermark, M., Nordgren, B., and S. Pink, "IP Header
      Compression", RFC 2507, February 1999.

[17] Casner, S. and V. Jacobson, "Compressing IP/UDP/RTP Headers for
      Low-Speed Serial Links", RFC 2508, February 1999.

[18] Degermark, M., "Requirements for robust IP/UDP/RTP header
      compression", RFC 3096, July 2001.

[19] Koren, T., Casner, S., Geevarghese, J., Thompson, B., and P.
      Ruddy, "Enhanced Compressed RTP (CRTP) for Links with High
      Delay, Packet Loss and Reordering", RFC 3545, July 2003.

[20] Degermark, M., Hannu, H., Jonsson, L.E., and K. Svanbro,
      "Evaluation of CRTP Performance over Cellular Radio Networks",
      IEEE Personal Communication Magazine, Volume 7, number 4, pp.
      20-25, August 2000.

[21] IANA registry, "RObust Header Compression (ROHC) Profile
      Identifiers", http://www.iana.org/assignments/rohc-pro-ids

[22] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA
      Considerations Section in RFCs", BCP 26, RFC 2434, October 1998.

Appendix A. CRC Algorithm

```perl
#!/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);
```

Authors' Addresses

    Lars-Erik Jonsson
    Optand 737
    SE-831 92 Ostersund, Sweden

    Phone: +46 70 365 20 58
    EMail: lars-erik@lejonsson.com

    Ghyslain Pelletier
    Ericsson AB
    Box 920
    SE-971 28 Lulea, Sweden

    Phone: +46 8 404 29 43
    Fax: +46 920 996 21
    EMail: ghyslain.pelletier@ericsson.com

    Kristofer Sandlund
    Ericsson AB
    Box 920
    SE-971 28 Lulea, Sweden

    Phone: +46 8 404 41 58
    Fax: +46 920 996 21
    EMail: kristofer.sandlund@ericsson.com

Full Copyright Statement

Intellectual Property

Acknowledgement