Network Working Group
INTERNET-DRAFT
Expires: December 2000

Lars-Erik Jonsson, Ericsson
Mikael Degermark, Lulea University
Hans Hannu, Ericsson
Krister Svanbro, Ericsson
Sweden

June 15, 2000

ROBust Checksum-based header COmpression (ROCCO)

<draft-ietf-rohc-rtp-rocco-01.txt>

ROCCO Version: 06

Status of this memo

This document is a submission of the IETF ROHC WG. Comments should be directed to its mailing list, rohc@cdt.luth.se.

Abstract

Existing header compression schemes do not work well when used over links with significant error rates, especially when the round-trip time of the link is long. For many bandwidth limited links where header compression is essential, such characteristics are common.

A header compression framework and a highly robust and efficient header compression scheme is introduced in this document, adaptable to the characteristics of the link over which it is used and also to the properties of the packet streams it compresses.

Table of contents

Document history

The original name of this internet draft was draft-jonsson-robust-hc. Since it now has been submitted as a ROHC WG draft, the name has changed and then also the numbering. However, the old draft version numbering is kept as a version reference for the protocol and this one should be referred to as ROCCO 06.

00	1999-06-22	First release.
01	1999-09-01	Only small corrections and modifications. Cut-and-paste errors from the 00 draft removed.
02	1999-10-22	Generalized concept with a number of different profiles. New chapters added describing profile negotiation, implementation status and security.
03	2000-01-18	LSP encoding and one-octet profiles introduced. Modified and simplified extension formats and small changes in the CONTEXT_REQUEST packets.
04	2000-03-10	The CONTEXT_UPDATE packet has changed its name to DYNAMIC while also being slightly modified. Both the STATIC and the DYNAMIC packet now include a header compression CRC of 8 bits to ensure reliability of the scheme. Some profiles have been modified, some renumbered, some removed and some added. The CONTEXT_REQUEST packet has been replaced by a more general FEEDBACK packet type with several sub-types including three that leaves much room for implementation features. The profile definitions have been improved in many ways with more details and clarifications. New chapters have been added discussing implementation issues and possible further work.
05	2000-05-24	New, additional compressed header formats have been added, both for timer-based timestamp decompression and for the cases without that functionality. The extension formats have also been updated. Initial chapters have been reorganized to get a better structure. CID-field has been moved to the beginning of each header. Minor changes have been applied to various parts of the specification.
06	2000-06-15	Errors in the previous version have been corrected. Since the ROHC group is now creating a final robust header compression scheme, ROCCO has served its

purposes and will be terminated. This is therefore
the final ROCCO version.

1. Introduction

During the last five years, two communication technologies in particular have become commonly used by the general public: cellular telephony and the Internet. Cellular telephony has provided its users with the revolutionary possibility of always being reachable with reasonable service quality no matter where they are. However, until now the main service provided has been speech. With the Internet, the conditions have been almost the opposite. While flexibility for all kinds of usage has been its strength, its focus has been on fixed connections and large terminals, and the experienced quality of some services (such as Internet telephony) has generally been low.

Today, IP telephony is gaining momentum thanks to improved technical solutions. It seems reasonable to believe that in the years to come, IP will become a commonly used way to carry telephony. Some future cellular telephony links might also be based on IP and IP telephony. Cellular phones may have IP stacks supporting not only audio and video, but also web browsing, email, gaming, etc.

The scenario we are envisioning might then be the one in Figure 1.1, where two mobile terminals are communicating with each other. Both are connected to base stations over cellular links, and the base stations are connected to each other through a wired (or possibly wireless) network. Instead of two mobile terminals, there could of course be one mobile and one wired terminal, but the case with two cellular links is technically more demanding.

```
Mobile            Base                      Base            Mobile
Terminal          Station                   Station         Terminal


      |  ~   ~   ~  \ /                    \ /  ~   ~   ~   ~  |
      |              |                      |                  |
   +--+              |                      |               +--+
   |  |              |                      |               |  |
   |  |              |                      |               |  |
   +--+              |                      |               +--+
                     |                      |
                     |======================|

         Cellular              Wired               Cellular
         Link                  Network             Link
```

Figure 1.1 : Scenario for IP telephony over cellular links

It is obvious that the wired network can be IP-based. With the cellular links, the situation is less clear. IP could be terminated in the fixed network, and special solutions implemented for each supported service over the cellular link. However, this would limit the flexibility of the services supported. If technically and economically feasible, a solution with pure IP all the way from terminal to terminal would have certain advantages. However, to make IP-all-the-way a viable alternative, a number of problems have to be addressed, especially regarding bandwidth efficiency.

For cellular phone systems, it is of vital importance to use the scarce radio resources in an efficient way. A sufficient number of users per cell is crucial, otherwise deployment costs will be prohibitive [CELL]. The quality of the voice service should also be as good as in today's cellular systems. It is likely that even with support for new services, lower quality of the voice service is acceptable only if costs are significantly reduced.

A problem with IP over cellular links when used for interactive voice conversations is the large header overhead. Speech data for IP telephony will most likely be carried by RTP [RTP]. A packet will then, in addition to link layer framing, have an IP [IPv4] header (20 octets), a UDP [UDP] header (8 octets), and an RTP header (12 octets) for a total of 40 octets. With IPv6 [IPv6], the IP header is 40 octets for a total of 60 octets. The size of the payload depends on the speech coding and frame sizes used and may be as low as 15-20 octets.

From these numbers, the need for reducing header sizes for efficiency reasons is obvious. However, cellular links have characteristics that make header compression as defined in [IPHC,CRTP,PPPHC] perform less than well. The most important characteristic is the lossy behavior of cellular links, where a bit error rate (BER) as high as 1e-3 must be accepted to keep the radio resources efficiently utilized [CELL]. In severe operating situations, the BER can be as high as 1e-2. The other problematic characteristic is the long round-trip time (RTT) of the cellular link, which can be as high as 100-200 milliseconds [CELL]. A viable header compression scheme for cellular links must be able to handle loss on the link between the compression and decompression point as well as loss before the compression point.

Bandwidth is the most costly resource in cellular links. Processing power is very cheap in comparison. Implementation or computational simplicity of a header compression scheme is therefore of less importance than its compression ratio and robustness.

2. Terminology

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

BER

Bit Error Rate. Cellular radio links have a rather high BER. In this document BER is usually given as a frequency, but one also needs to consider the error distribution as bit errors are not independent. In our simulations we use a channel with a certain BER, and the error distribution is according to a realistic channel [WCDMA].

Cellular links

Wireless links between mobile terminals and base stations. The BER and the RTT are rather high in order to achieve an efficient system overall.

Compression efficiency

The performance of a header compression scheme can be described with three parameters, compression efficiency, robustness and compression reliability. The compression efficiency is determined by how much the header sizes are reduced by the compression scheme.

Compression reliability

The performance of a header compression scheme can be described with three parameters, compression efficiency, robustness and compression reliability. The compression reliability is a measure for how well the scheme ensures that the decompressed headers are not erroneous and the possibility to avoid error propagation from the decompressor.

Context

The context is the state which the compressor uses to compress a header and which the decompressor uses to decompress a header. The context is basically the uncompressed version of the last header sent (compressor) or received (decompressor) over the link, except for fields in the header that are included "as-is" in compressed headers or can be inferred from, e.g., the size of the link-level frame. The context can also contain additional information describing the packet stream, for example the typical inter-packet increase in sequence numbers or timestamps.

Context damage

When the context of the decompressor is not consistent with the context of the compressor, header decompression will fail. This situation can occur when the context of the decompressor has not been initialized properly or when packets have been lost or damaged between compressor and decompressor. Packets which cannot be decompressed due to inconsistent contexts are said to be lost due to context damage.

Context repair mechanism

To avoid excessive context damage, a context repair mechanism is needed. Context repair mechanisms can be based on explicit requests for context updates, periodic updates sent by the compressor, or methods for local repair at the decompressor side.

FER

Frame Error Rate. The FER considered in this document includes the frames lost on the channel between compressor and decompressor and frames lost due to context damage. FER is here defined to be identical to packet loss rate.

Header compression profile

A header compression profile is a specification of how to compress the headers of a certain kind of packet stream over a certain kind of link. Compression profiles provide the details of the header compression framework introduced in this document. The profile concept makes use of profile identifiers to separate different profiles which are used when setting up the compression scheme. All variations and parameters of the header compression scheme are handled by different profile identifiers, which makes the number of profiles rather large. This can act as a deterrent when first studying the concept, but is a real strength for several reasons. One advantage of this merging of parameters into one is that new parameters can be added by the endpoints without affecting the negotiation requirements on the link in between. Another benefit of the concept is that different combinations of functionality might be implemented with different methods, meaning that the scheme can be optimized regardless of what functionality is enabled. Finally, it should be noted that even if there are a large number of profiles, only a small number of them can/will be implemented over a specific link (IPv4 and IPv6 profiles will for example probably not coexist). Most profiles usable in a certain environment will probably also be almost identical from an implementation point of view.

Header compression CRC

A CRC (Cyclic Redundancy Checksum) computed by the compressor and included in each compressed header. Its main purpose is to provide a way for the decompressor to reliably verify the correctness of reconstructed headers. What values the CRC is computed over depends on the packet type it is included in; typically it covers most of the original header fields.

Pre-HC links

Pre-HC links are all links before the header compression point. If we consider a path with cellular links as first and last hops, the Pre-HC links for the compressor at the last link are the first cellular link plus the wired links in between.

Robustness

The performance of a header compression scheme can be described with three parameters, compression efficiency, robustness and compression reliability. A robust scheme tolerates errors on the link over which header compression takes place without losing additional packets, introducing additional errors, or using more bandwidth.

RTT

Round Trip Time - The time it takes to send a packet back and forth over the link.

Simplex link

A simplex (or unidirectional) link is a point to point link without a return channel. Over simplex links, header compression must rely on periodic refreshes since feedback from the decompressor can not be sent to the compressor. For simplex links, a header compression CRC is mandatory to guarantee correct decompression.

Spectrum efficiency

Radio resources are limited and expensive. Therefore they must be used efficiently to make the system economically feasible. In cellular systems this is achieved by maximizing the number of users served within each cell, while the quality of the provided services is kept at an acceptable level. A consequence of efficient spectrum use is a high BER, even after channel coding with error correction.

Timestamp delta

The timestamp delta is the increase in the timestamp value between two consecutive packets.

3. Background

This chapter provides a background to the subject of header compression. The fundamental ideas are described together with descriptions of existing header compression schemes, their drawbacks and requirements and motivation for new header compression solutions.

3.1. Header compression fundamentals

The main reason why header compression can be done at all is the fact that there are lots of redundancy between header fields, both within the same packet header but especially between consecutive packets belonging to the same packet stream. By sending static field information only initially and utilize dependencies and predictability's for other fields, the header size can be significantly reduced for most packets.

In general, header compression methods maintain a context, which is essentially the uncompressed version of the last header sent over the link, at both compressor and decompressor. Compression and decompression are done relative to the context. When compressed headers carry differences from the previous header, each compressed header will update the context of the decompressor. When a packet is lost between compressor and decompressor, the context of the decompressor will be brought out of sync since it is not updated correctly. A header compression method must have a way to repair the context, i.e. bring it into sync, after such events.

3.2. Existing header compression schemes

The original header compression scheme, CTCP [VJHC], was invented by Van Jacobson. CTCP compressed the 40 octet IP+TCP header to 4 octets.

The CTCP compressor detects transport-level retransmissions and sends a header that updates the context completely when they occur. This repair mechanism does not require any explicit signaling between compressor and decompressor.

CRTP [CRTP, IPHC] by Casner and Jacobson is a header compression scheme that compresses 40 octets of IPv4/UDP/RTP headers to a minimum of 2 octets when no UDP checksum is present. If the UDP checksum is present, the minimum CRTP header is 4 octets. CRTP cannot use the same repair mechanism as CTCP since UDP/RTP does not retransmit. Instead, CRTP uses explicit signaling messages from decompressor to compressor, called CONTEXT_STATE messages, to indicate that the context is out of sync. The link roundtrip time will thus limit the speed of this context repair mechanism.

On lossy links with long roundtrip times, such as most cellular links, CRTP does not perform well. Each lost packet over the link causes several subsequent packets to be lost since the context is out of sync during at least one link roundtrip time. This behavior is documented in [CRTPC]. For voice conversations such long loss events will degrade the voice quality. Moreover, bandwidth is wasted by the large headers sent by CRTP when updating the context. [CRTPC] found that CRTP performed much worse than ideally for a lossy cellular link. It is clear that CRTP alone is not a viable header compression scheme for cellular links.

To avoid losing headers due to the context being out of sync, CRTP decompressors can attempt to repair the context locally by using a mechanism known as TWICE. Each CRTP packet contains a counter which is incremented by one for each packet sent out by the CRTP compressor. If the counter increases by more than one, at least one packet was lost over the link. The decompressor then attempts to repair the context by guessing how the lost packet(s) would have updated it. The guess is then verified by decompressing the packet and checking the UDP checksum - if it succeeds, the repair is deemed successful and the packet can be forwarded or delivered. TWICE has got its name from the observation that when the compressed packet stream is regular, the correct guess is to apply the update in the current packet twice. [CRTPC] found that even with TWICE, CRTP doubled the number of lost packets. TWICE improves CRTP performance significantly. However, there are several problems with using TWICE:

1) It becomes mandatory to use the UDP checksum:

   - the minimal compressed header size increases by 100% to 4 octets.

   - most speech codecs developed for cellular links tolerate errors in the encoded data. Such codecs will not want to enable the UDP checksum, since they want damaged packets to be delivered.

   - errors in the payload will make the UDP checksum fail when the guess is correct (and might make it succeed when it is wrong).

2) Loss in an RTP stream that occurs before the compression point will make updates in CRTP headers less regular. Simple-minded versions of TWICE will then perform badly. More sophisticated versions would need more repair attempts to succeed.

3.3. Requirements on a new header compression scheme

The major problem with CRTP is that it is not sufficiently robust against packets being damaged between compressor and decompressor. A viable header compression scheme must be less fragile. This increased robustness must be obtained without increasing the compressed header size; a larger header would make IP telephony over cellular links economically unattractive.

A major cause of the bad performance of CRTP over cellular links is the long link roundtrip time, during which many packets are lost when the context is out of sync. This problem can be attacked directly by finding ways to reduce the link roundtrip time. Future generations of cellular technologies may indeed achieve lower link roundtrip times. However, these will probably always be rather high [CELL]. The benefits in terms of lower loss and smaller bandwidth demands if the context can be repaired locally will be present even if the link roundtrip time is decreased. A reliable way to detect a successful context repair is then needed.

One might argue that a better way to solve the problem is to improve the cellular link so that packet loss is less likely to occur. It would of course be nice if the links were almost error free, but such a system would not be able to support a sufficiently large number of users per cell and would thus be economically unfeasible [CELL].

One might also argue that the speech codecs should be able to deal with the kind of packet loss induced by CRTP, in particular since the speech codecs probably must be able to deal with packet loss anyway if the RTP stream crosses the Internet. While the latter is true, the kind of loss induced by CRTP is difficult to deal with. It is usually not possible to hide a loss event where well over 100 ms worth of sound is completely lost. If such loss occurs frequently at both ends of the path, the speech quality will suffer.

3.4. Classification of header fields

As mentioned earlier, header compression is possible due to the fact that there are much redundancy between header field values within packets, but especially between consecutive packets. To utilize these properties for header compression, it is important to understand the behavior of the various header fields. To do this, all header fields have been classified in detail in appendix A. The fields are first classified on a high level and then some of them are studied more in detail. Finally, the appendix concludes with recommendations about how the various fields should be handled by header compression algorithms. The main conclusion that can be drawn is that most of the header fields can easily be compressed away since they are never or seldom changing. Only 5 fields with a total size of about 10 octets are rather difficult to compress and must be handled in a sophisticated way by the compression scheme. Those fields are:

- IPv4 Identification (16 bits)
- UDP Checksum (16 bits)
- RTP Marker (1 bit)
- RTP Sequence Number (16 bits)
- RTP Timestamp (32 bits)

It is rather obvious that these field then will have a large impact on how a header compression scheme is designed. However, all details about this should be found in Appendix A.

4. Header compression framework

A solution is proposed for efficient, robust and reliable header compression, ROCCO. The scheme is heavily geared towards local repair of the context in combination with robust encoding of header fields. What is needed is a reliable way to detect when a repair attempt has succeeded, plus possibly hints to the decompressor about how the header fields have changed.

The key element of the proposed header compression scheme is that compressed headers should carry a CRC computed over the header before compression. This provides a reliable way to detect whether decompression and context repair have succeeded. In addition to the CRC, the header could contain codes and additional information as needed for decompression.

A completely general solution cannot achieve compression rates high enough to make IP telephony over cellular economically feasible. On the other hand, the solution needs to be extendable so that other kinds of packet streams can also be compressed well. Therefore, we envision a scheme where the basic framework is supplemented with a set of compression profiles, where each compression profile provides the exact details on how a packet stream is to be compressed and decompressed. The use of compression profiles allows the basic framework to be adapted to the properties of packet streams as well as various link properties.

The ROCCO header compression framework does not state any exact details about how the compression is to be performed, what formats the headers should have, etc. This is left to the compression profiles to define. The framework instead describes general principles for how to do header compression "the ROCCO way". The header compression profile concept is presented describing what a profile is, what to consider when designing a profile and what every profile must or should define. The concept also exactly states the rules regarding negotiation of compression profiles.

4.1. General ROCCO principles

ROCCO header compression is based on the principle of decompressor context repair attempts relying on a header compression CRC included in compressed headers. Profiles will define various packet types and all of them do not have to carry a header compression CRC. In general, if the CRC is present it is RECOMMENDED to calculate it over the uncompressed header, but profiles MAY define the coverage differently for some packet types.

Distinguishing packet streams and packet types is necessary, but some of that information may be available from the underlying technology. To avoid wasting precious header bits, it is left to the compression profile to decide how to distinguish packet types and packet streams. This allows efficient use of header bits overall.

If each packet stream has its own logical channel, it is not necessary to have any additional information for distinguishing between streams. Otherwise there MUST be slightly different profiles defined with support for various numbers of concurrent packet streams.

The link layer could carry explicit information about packet types, but that would not lead to an efficient use of bits, since different profiles could use different number of packet types. If the packet type distinguishing mechanism is included in the header compression profile instead, the profile could optimize the bit usage of that mechanism to its own packet types. However, it is up to the profile designer to choose if this mechanism is included in the profile or required from the link layer.

A compression profile MAY define headers which do not have a corresponding original packet. Such packets would be internal header compression packets, and would not be delivered further from the decompressor. An example would be to initially send non-changing fields of a packet stream as a separate packet. Another example would be to send packets to update the RTP timestamp field even when no RTP packets arrive, in order to decrease the increment in the RTP timestamp field when a packet does arrive.

The profiles defined in this document SHOULD be considered as examples for how profiles are supposed to be defined and described.

4.2. Data structures

The compression scheme needs to maintain a context for compression and decompression of a packet stream. The context must be kept updated at both compressor and decompressor. The context is essentially the header of the last packet transmitted, and includes all static header fields plus some fields that change more or less frequently. If the compression profile used is designed to handle a certain amount of packet loss on the link, both compressor and decompressor will typically keep information about earlier packets; packets that arrived before the current packet. Finally, there may be packet stream related information such as field deltas (e.g. RTP timestamp) or a list of which CSRC items have occurred in the packet stream.

4.3. Header compression profiles

The details on how a packet stream is to be compressed and decompressed are determined by a compression profile. Over a link a number of profiles can be active, either within the same or within different logical channels. How to determine what profile to use for a certain packet stream is not defined in this document, but the usage MUST be negotiated between compressor and decompressor as described in the subsequent chapter.

One way to select a suitable profile is to have a common channel over which a general-purpose header compression profile is used. When the packet stream characteristics are identified, it is switched to another channel where a suitable compression profile is used.

Profiles can be defined to compress one packet stream only, in which case the link layer must be able to separate packet streams. Profiles can also be defined for compression of more than one packet stream, in which case the profile must provide a way to distinguish between the packet streams.

Important parameters to consider when designing a profile are:

- what kind of packet streams to compress (IPv6, IPv4, UDP, UDP/RTP, TCP) and if UDP, whether the UDP checksum is supported.

- the rate and pattern of loss of the channel.

- the pattern of change of the changing fields.

- the expected rate and pattern of loss and reordering before the compression point.

- if included in the profile, the number of concurrent packet streams supported through context identifiers.

- what support there is from the link layer, such as the number of packet streams supported, and if it can indicate packet types.

When these things have been considered, the specifics of the profile can be determined. The profile MUST specify:

- the exact semantics of the various packet types and how the desired functionality is supported.

- the size of, polynomials for, and what the Header Compression CRC covers for all packet types.

- the information needed in the contexts for compression and decompression, including history information and properties of the packet stream.

- procedures for compression and decompression.

- how compression is initiated (which packets are used and how).

- description of context repair mechanisms.

Chapter 5 defines compression profiles optimized for conversational voice traffic over cellular radio links.

4.4. **Profile negotiation**

To initiate ROCCO header compression, compressor and decompressor must be able to negotiate which header compression profile to use. A header compression profile is identified by a 16 bit profile identifier, and underlying link layers MUST provide a way to negotiate this.

4.5. **Link layer requirements**

This chapter lists general ROCCO requirements on an underlying link layer. Profiles could also state additional requirements, but these MUST be provided for ALL ROCCO profiles. See also [LLG].

Framing

Framing, which makes it possible to separate different packets, is the most important link layer functionality.

Length

Most link layers can indicate the length of the packet, and this information has therefore been removed from the packet headers. This means that it now MUST be given by the link layer.

Profile negotiation

In addition to the packet handling mechanisms above, the link layer MUST also provide a way to carry on the negotiation of header compression profiles, described in chapter 5.4.

5. Header compression profiles optimized for voice traffic

This section exemplifies how the framework outlined in chapter 4 could be instantiated by defining profiles optimized for header compression of packet streams carrying voice data. A number of profiles are defined providing support for both IPv6 and IPv4 in combination with various functionality.

5.1. Usage scenarios, environment and requirements

These profiles are optimized for voice traffic over error prone links. The profiles are designed to successfully handle loss of several consecutive packets over the link, without introducing any additional loss. Packet type identification is included in all profiles, which means that such functionality SHOULD NOT be provided by the link layer used. Regarding packet stream separation, various profiles are defined supporting different numbers of concurrent streams.
As an error prone link with similar characteristics is expected at the other end of the connection (see Figure 1.1), the profiles are also designed to handle some consecutive lost packet before the compression point without increasing the size of the compressed header. The profiles are also in general designed to handle reordering of single packets before the compression point without increasing the size of compressed headers.

5.2. Profile definitions

This document defines a number of different header compression profiles. The definitions are built up of the requirements on and capabilities of each profile in combination with information about which mechanisms are used to implement the desired behavior.

5.2.1. List of defined profiles

All defined profiles are listed in Table 5.1 together with their characteristics, capabilities and pointers to implementation details that may differ from profile to profile. For more information about the profile concept see chapter 4.3 and "Header compression profile" in the Terminology chapter.

The first seven columns state requirements on and capabilities of the profiles. The meaning of the columns are:

Nr This is the identification number for each profile. These numbers are used when negotiating profiles in the header compression setup phase. The numbers are preliminary and will

perhaps change in future versions of this profile
specification.

IPv This is the IP version for which the profile is designed. Possible values for this column are 6 and 4.

CID This column gives the number of concurrent packet streams that are supported by the header compression profile through context identifiers (CIDs).

Chk This column indicates whether the profile supports packet streams with the UDP checksum (E)nabled or D(isabled).

ID For profiles supporting IPv4, this column indicates which behavior of the IPv4 Identification field the profile is optimized for. Possible values in this column are:

(S)EQUENTIAL These profiles can handle all kind of Identification assignment methods but will be less efficient than RANDOM profiles if the assignment truly is random. If the value is sequentially assigned, no extra overhead is added for Identification.

(R)ANDOM These profiles are recommended if it is known that random assignment is used. The Identification field will be included "as-is" which means that the header size will increase by two octets.

TbT Timer-based Timestamp decompression. Requires a timer at the decompressor side to estimate timestamp jumps. Compressor never sends more than a few bits of timestamp LSB with these profiles. Can be (E)nabled or (D)isabled (see chapter 5.5.3).

S S gives the minimal header Size for the profile.

The next five columns indicate how each profile is implemented. This includes header formats for STATIC (STA, see chapter 5.4.1), DYNAMIC (DYN, see chapter 5.4.2) and COMPRESSED (COM, see chapter 5.4.3) packets, and also what EXTENSION (EXT, see chapter 5.4.4) formats are used with the COMPRESSED packets. The CRC column tells the coverage of the header compression CRC: uncompressed (H)eader or the same coverage as for the UDP (C)hecksum (see chapter 5.6).

Nr	IPv	CID	Chk	ID	TbT	S	STA	DYN	COM	EXT	CRC
1	6	1	E	-	E	2	1	1	1	A	C
2	6	1	E	-	D	2	1	1	1	A	C
3	6	256	E	-	E	3	2	2	2	A	C
4	6	256	E	-	D	3	2	2	2	A	C
5	4	1	D	S	E	1	3	3	5, 9	D	H
6	4	1	D	S	E	2	3	3	1	D	H
7	4	1	D	S	D	1	3	3	5, 13	B	H
8	4	1	D	S	D	2	3	3	1	B	H
9	4	1	D	R	E	3	3	3	7, 11	C	H
10	4	1	D	R	E	4	3	3	3	C	H
11	4	1	D	R	D	3	3	3	7, 15	A	H
12	4	1	D	R	D	4	3	3	3	A	H
13	4	1	E	S	E	2	3	5	1	D	C
14	4	1	E	S	D	2	3	5	1	B	C
15	4	1	E	R	E	4	3	5	3	C	C
16	4	1	E	R	D	4	3	5	3	A	C
17	4	256	D	S	E	2	4	4	6, 10	D	H
18	4	256	D	S	E	3	4	4	2	D	H
19	4	256	D	S	D	2	4	4	6, 14	B	H
20	4	256	D	S	D	3	4	4	2	B	H
21	4	256	D	R	E	4	4	4	8, 12	C	H
22	4	256	D	R	E	5	4	4	4	C	H

```
    +-----+---+-----+---+---+---+  +---+  +---+---+-------+---+---+
    | 23  | 4 | 256 | D | R | D |  | 4 |  | 4 | 4 | 8, 16 | A | H |
    +-----+---+-----+---+---+---+  +---+  +---+---+-------+---+---+
```

Nr	IPv	CID	Chk	ID	TbT	S	STA	DYN	COM	EXT	CRC
24	4	256	D	R	D	5	4	4	4	A	H
25	4	256	E	S	E	3	4	6	2	D	C
26	4	256	E	S	D	3	4	6	2	B	C
27	4	256	E	R	E	5	4	6	4	C	C
28	4	256	E	R	D	5	4	6	4	A	C

Table 5.1 : List of defined profiles

5.2.2. Additional common profile characteristics

In addition to what was stated in the left part of Table 5.1, the following applies to all the profiles defined in this document:

Packet stream characteristics

These profiles are designed for packet streams carrying conversational voice data.

Link layer requirements

Except for the general link layer requirements (framing, length & profile negotiation) stated in chapter 4.5, these profiles also require a reliable link layer CRC covering at least the header part of the packet. The CRC SHOULD ensure that packets with errors in the header part are never delivered.

Pre link characteristics

With these profiles, several consecutive packet losses before the header compression point are handled without introducing additional header overhead. Packet reordering on pre links is expected to be uncommon but is handled, very efficiently when limited.

Link Characteristics

The link over which header compression is performed is expected to have a loss characteristic that very seldom leads to loss of many consecutive packets. These profiles can without extra reconstruction attempts handle loss of up to 25 consecutive packets over the link without losing context, and even if loss on pre links decreases this robustness, it should be more than sufficient for all realistic scenarios. The round-trip time of the link is expected to be between 100 and 200 milliseconds.

5.3. Encoding methods used

The analysis of header field changes in appendix A excluded changes due to loss and/or reordering before the header compression point. Such changes will have an impact on the regularity of the RTP sequence number, the RTP timestamp value and, for IPv4, the IP ID value. However, as described in A.2, both the RTP timestamp and the IP ID value (if sequentially assigned) are expected to follow the RTP sequence number for most packets. The most important task is then to communicate RTP sequence number information in an efficient way. The profiles defined in this document make use of two different methods of handling the sequence number field and also other fields, LSB encoding and LSP encoding. The methods are interpreted in different ways for different fields, and this chapter therefore describes these methods in a general way. How these two methods are applied to fields in different compressed headers is described in chapter 5.6.

5.3.1. Least Significant Bits (LSB) encoding

A commonly used method for updating fields whose values are always subject to small changes (usually positive) is Least Significant Bits (LSB) encoding. For example, an increase of up to 16 could be handled with only 4 bits with LSB encoding (if decreases are not expected). This method is used for many different fields by the ROCCO profiles defined in this document, especially when information such as timestamps is sent in EXTENDED COMPRESSED headers. If a field is labeled "<fieldname> LSB", it means that the field contains only the least significant bits of the corresponding original field.

5.3.2. Least Significant Part (LSP) encoding

One restriction with LSB encoding is that whole bits are needed, meaning that only 2, 4, 8, 16, 32, ... code-points could be used. In some cases, especially when several mechanisms are integrated for efficiency reasons, it would be desirable to have a method that could make use of any number of available code-points. To signal one special event one could either use one single bit or, if the event is not to be signaled in parallel with other information, as one bit pattern for several bits. That would leave more bit patterns for other usage.

Assume that we have 4 special events to signal and 5 bits available.

Taking 2 bits for these events, then there would be 3 bits (8 code-points) left for other usage. If we instead reserved 4 of the code-points represented by all 5 bits, there would instead be 32-4=28 code-points left for other usage. The only disadvantage would be that the bits cannot be used for both purposes at the same time.

What would be desirable is to do LSB encoding of code-points instead of whole bits. Therefore the method called Least Significant Part (LSP) encoding has been introduced. LSP encoding of size (number of code-points) M for a value N is defined as:

```
LSP:M(N) = N modulo M
```

An example showing the LSP encoding and decoding of a counter S(n) with M code-points is used below to illustrate the LSP principle. S'(n) is the decoded value corresponding to the original S(n) value. With S'(n-1), we denote the last correctly decompressed value.

```
Input sequence: S(n)
Encoded sequence: LSP:M(S(n)) = S(n) modulo M
Decoded sequence: S'(n) = S'(n-1) - LSP:M(S'(n-1)) + LSP:M(S(n)) =
                          S'(n-1) - S(n-1) modulo M + S(n) modulo M
```

To handle modulo wrap-around, an additional verification is inserted. If the decoded value S'(n) is smaller than S'(n-1)-R, S'(n) is increased with M (reordering of order R is then handled with this encoding).

When applying LSP encoding, there are thus two parameters that must be set:

```
M - The number of code-points to use (modulo value)
R - The reordering order to handle
```

A similar mechanism as for modulo wrap-around should also be used to handle full-field wrap-around.

5.3.3. LSB or LSP encoding with extended range

If needed, it could be good to extend the range covered by the LSB or LSP encoding. For the LSB case, it is simple to send only the next more significant bits. For the LSP, what must be done is to rewrite the definition of LSP so that it defines an additional parameter.

The LSP definition from previous chapter can instead be expressed as:

```
LSP:M(N) = N - INT:M(N)*M      [ INT:M(N) = (N - LSP:M(N)) / M) ]
```

And in that case, INT:M(N) is the integer part left after division. If additional bits can be transmitted to increase the range covered, this can be done by sending the least significant bits (LSB) of this integer part, INT:M(N). The example from previous chapter will then change into:

```
Input sequence: S(n)
Encoded sequence: LSP:M(S(n)) = S(n) modulo M
                  INT:M(S(n)) = (S(n)-LSP:M(S(n))) / M
Decoded sequence: S'(n) = S'(n-1) - LSP:M(S'(n-1)) * M
                                  + LSP:M(S(n)) * M
                                  - LSB(INT:M(S'(n-1))) * M
                                  + LSB(INT:M(S(n))) * M
```

5.4. Packet formats

The profiles defined in this document make use of four different packet types: STATIC, DYNAMIC, COMPRESSED and FEEDBACK.

To identify packet types, 4 bit patterns for the initial 5 bits of the first octet (not including a potential CID field) in all packets are reserved. These patterns are:

```
STATIC      11111
DYNAMIC     1110*    (both 11100 and 11101 are reserved for this)
FEEDBACK    11110
```

The other 28 (32-4) bit patterns indicate a COMPRESSED packet format and the usage of these patterns are explained further on.

This section defines the header formats of the four ordinary packet types STATIC, DYNAMIC, COMPRESSED and FEEDBACK together with descriptions of when and how to use them. A subsections is also dedicated to the EXTENSION formats of COMPRESSED headers.

5.4.1. Static information packets, initialization

The STATIC packet type is a packet containing no payload but only the header fields that are expected to be constant throughout the lifetime of the packet stream (classified as STATIC in appendix A). A packet of this kind MUST be sent once as the first packet from compressor to decompressor and also when requested by the decompressor (see chapter 5.4.5). The packet formats are shown below for IPv6 and IPv4, respectively. Note that some fields are only present in some of the STATIC packet types.

IPv6 (45-46 octets): STATIC1, STATIC2:

```
  0   1   2   3   4   5   6   7
+...+...+...+...+...+...+...+...+
:   Context Identifier  (CID)   :   only present in STATIC2
+---+---+---+---+---+---+---+---+
| 1   1   1   1   1 | - | - | - |
+---+---+---+---+---+---+---+---+
|                               |
+           Flow Label          +
|                               |
+               +---+---+---+---+
|               | - | - | P | E |
+---+---+---+---+---+---+---+---+
|                               |
/         Source Address        /   16 octets
|                               |
+---+---+---+---+---+---+---+---+
|                               |
/       Destination Address     /   16 octets
|                               |
+---+---+---+---+---+---+---+---+
|                               |
+           Source Port         +
|                               |
+---+---+---+---+---+---+---+---+
|                               |
+        Destination Port       +
|                               |
+---+---+---+---+---+---+---+---+
|                               |
/              SSRC             /   4 octets
|                               |
+---+---+---+---+---+---+---+---+
|    Header Compression CRC     |   see chapter 5.6.1.
+---+---+---+---+---+---+---+---+
```

IPv4 (18-19 octets): STATIC3, STATIC4:

```
       0   1   2   3   4   5   6   7
     +...+...+...+...+...+...+...+...+
     :   Context Identifier  (CID)   :   only present in STATIC4
     +---+---+---+---+---+---+---+---+
     | 1   1   1   1   1 | F | P | E |
     +---+---+---+---+---+---+---+---+
     |                               |
     /        Source Address         /   4 octets
     |                               |
     +---+---+---+---+---+---+---+---+
     |                               |
     /      Destination Address      /   4 octets
     |                               |
     +---+---+---+---+---+---+---+---+
     |                               |
     +          Source Port          +
     |                               |
     +---+---+---+---+---+---+---+---+
     |                               |
     +       Destination Port        +
     |                               |
     +---+---+---+---+---+---+---+---+
     |                               |
     /             SSRC              /   4 octets
     |                               |
     +---+---+---+---+---+---+---+---+
     |    Header Compression CRC     |   see chapter 5.6.1.
     +---+---+---+---+---+---+---+---+
```

All fields except for the initial five bits, the padding (-) and the Header Compression CRC are the ordinary IP, UDP and RTP fields (F=IPv4 May Fragment, P=RTP Padding, E=RTP Extension).

The number of STATIC packets sent on each occasion should be limited. If the decompressor receives DYNAMIC or COMPRESSED headers without having received a STATIC packet, the decompressor MUST send a STATIC_FAILURE_FEEDBACK packet.

5.4.2. Dynamic information packets

The DYNAMIC packet type has a header containing all changing header fields in their original, uncompressed form, and carries a payload just like ordinary COMPRESSED packets. This packet type is used after the initial STATIC packet to set up the decompressor context for the first time, and also whenever the header field information cannot be encoded in EXTENDED_COMPRESSED packets. DYNAMIC packets could be used due to significant field changes or upon INVALID_CONTEXT_FEEBACK.

All fields except for the initial four bits, the Timestamp Delta, and the Header Compression CRC are ordinary IP, UDP and RTP fields. The Timestamp Delta is the current delta between RTP timestamps in consecutive RTP packets. Initially this value SHOULD be set to 160.

The packet formats are shown below for IPv6 and IPv4, respectively. Note that some fields are only present in some of the DYNAMIC packet types.

IPv6 (13-16 octets + CSRC List of 0-60 octets): DYNAMIC1, DYNAMIC2:

```
  0   1   2   3   4   5   6   7
+...+...+...+...+...+...+...+...+
:   Context Identifier  (CID)   :  only in DYNAMIC2
+---+---+---+---+---+---+---+---+
| 1   1   1   0 |  CSRC Counter |
+---+---+---+---+---+---+---+---+
|         Traffic Class         |
+---+---+---+---+---+---+---+---+
|           Hop Limit           |
+---+---+---+---+---+---+---+---+
|                               |
+         UDP Checksum          +
|                               |
+---+---+---+---+---+---+---+---+
| M |       Payload Type        |
+---+---+---+---+---+---+---+---+
|                               |
+        Sequence Number        +
|                               |
+---+---+---+---+---+---+---+---+
|                               |
/           Timestamp           /  4 octets
|                               |
+---+---+---+---+---+---+---+---+
:                               :
:           CSRC List           :  0-15 x 4 octets
:                               :
+---+---+---+---+---+---+---+---+
|                               |
+        Timestamp Delta        +
|                               |
+---+---+---+---+---+---+---+---+
|    Header Compression CRC     |  see chapter 5.6.2.
+---+---+---+---+---+---+---+---+
/            Payload            /
```

```
      +---+---+---+---+---+---+---+---+
```

```
   IPv4 (15-18 octets + CSRC List of 0-60 octets): DYNAMIC3, DYNAMIC4,
                                                   DYNAMIC5, DYNAMIC6:

          0   1   2   3   4   5   6   7
        +...+...+...+...+...+...+...+...+
        :   Context Identifier  (CID)   :  only in DYNAMIC4 and DYNAMIC6
        +---+---+---+---+---+---+---+---+
        | 1   1   1   0 |  CSRC Counter |
        +---+---+---+---+---+---+---+---+
        |        Type Of Service        |
        +---+---+---+---+---+---+---+---+
        |                               |
        +        Identification         +
        |                               |
        +---+---+---+---+---+---+---+---+
        |          Time To Live         |
        +---+---+---+---+---+---+---+---+
        :                               :
        +          UDP Checksum         +  only in DYNAMIC5 and DYNAMIC6
        :                               :
        +---+---+---+---+---+---+---+---+
        | M |       Payload Type        |
        +---+---+---+---+---+---+---+---+
        |                               |
        +        Sequence Number        +
        |                               |
        +---+---+---+---+---+---+---+---+
        |                               |
        /           Timestamp           /  4 octets
        |                               |
        +---+---+---+---+---+---+---+---+
        :                               :
        :           CSRC List           :  0-15 x 4 octets
        :                               :
        +---+---+---+---+---+---+---+---+
        |                               |
        +            TS Delta           +
        |                               |
        +---+---+---+---+---+---+---+---+
        |    Header Compression CRC     |  see chapter 5.6.2.
        +---+---+---+---+---+---+---+---+
        /            Payload            /
        +---+---+---+---+---+---+---+---+
```

Each time a DYNAMIC packet is sent, several subsequent packets SHOULD also be DYNAMIC packets to ensure a successful update even when packets are lost. Context updates both with DYNAMIC and COMPRESSED packets could also be acknowledged with CONTEXT_UPDATED_FEEDBACK.

5.4.3. Compressed packets

The COMPRESSED packet type is the most commonly used packet and is designed to handle ordinary changes as efficiently as possible.

When changes are regular, all information is carried in the base header. When desired, it is possible to send additional information in extensions to the COMPRESSED base-header.

The COMPRESSED base-header formats are shown below. Note that some fields are only present in some of the COMPRESSED packet types.

Defines packet types: COMPRESSED1..COMPRESSED4:

```
  0   1   2   3   4   5   6   7
+...+...+...+...+...+...+...+...+
:   Context Identifier  (CID)   : only in COMPRESSED type 2 and 4
+---+---+---+---+---+---+---+---+
|   Sequence LSP#   |           |    # see chapter 5.5.2
+---+---+---+---+---+       +---+
|   Header Compression CRC* | X |    * see chapter 5.6.3
+---+---+---+---+---+---+---+---+
:                               :
+        Identification         + only in COMPRESSED type 3 and 4
:                               :
+...+...+...+...+...+...+...+...+
:                               :
/           Extension           / only present if X=1
:                               :
+---+---+---+---+---+---+---+---+
/             Payload           /
+---+---+---+---+---+---+---+---+
```

Defines packet types: COMPRESSED5..COMPRESSED8:

```
  0   1   2   3   4   5   6   7
+...+...+...+...+...+...+...+...+
:   Context Identifier  (CID)   : only in COMPRESSED 6 and 8
+---+---+---+---+---+---+---+---+
| 0 | Sequence LSB# |    CRC*   |    # see chapter 5.5.1
+---+---+---+---+---+---+---+---+    * see chapter 5.6.3
:                               :
+        Identification         + only in COMPRESSED 7 and 8
:                               :
+---+---+---+---+---+---+---+---+
/             Payload           /
+---+---+---+---+---+---+---+---+
```

Defines packet types: COMPRESSED9..COMPRESSED12:

```
     0   1   2   3   4   5   6   7
   +...+...+...+...+...+...+...+...+
   :   Context Identifier  (CID)   : only in COMPRESSED 10 and 12
   +---+---+---+---+---+---+---+---+
   | 1   0 |          CRC*         |    * see chapter 5.6.3
   +---+---+---+---+---+---+---+---+
   | Sequence LSB# |  STS LSB# | X |    # see chapter 5.5.1
   +---+---+---+---+---+---+---+---+
   :                               :
   +        Identification         + only in COMPRESSED 11 and 12
   :                               :
   +...+...+...+...+...+...+...+...+
   :                               :
   /           Extension           / only present if X=1
   :                               :
   +---+---+---+---+---+---+---+---+
   /            Payload            /
   +---+---+---+---+---+---+---+---+
```

Defines packet types: COMPRESSED13..COMPRESSED16:

```
     0   1   2   3   4   5   6   7
   +...+...+...+...+...+...+...+...+
   :   Context Identifier  (CID)   : only in COMPRESSED 14 and 16
   +---+---+---+---+---+---+---+---+
   | 1   0 | M |      STS LSB#     |    # see chapter 5.5.1
   +---+---+---+---+---+---+---+---+
   | Sequence LSB# |    CRC*   | X |    * see chapter 5.6.3
   +---+---+---+---+---+---+---+---+
   :                               :
   +        Identification         + only in COMPRESSED 15 and 16
   :                               :
   +...+...+...+...+...+...+...+...+
   :                               :
   /           Extension           / only present if X=1
   :                               :
   +---+---+---+---+---+---+---+---+
   /            Payload            /
   +---+---+---+---+---+---+---+---+
```

The coverage of the Header Compression CRC is described in chapter 5.6.3. In that chapter, the CRC polynomials to use are also defined.

The interpretations of the Sequence and STS (Scaled TimeStamp) fields for different profiles are given in section 5.5.

5.4.4. Extensions to compressed headers

Less regular changes in the header fields or updates of decompressor contexts require an extension in addition to the base header. When there is an extension present in the COMPRESSED packet, this is indicated by the extension bit (X) being set. Extensions are of variable size depending on the information needed to be transmitted. However, the first three extension bits are used as an extension Type field for all extension formats. The extension can carry an M-bit, a t-bit, a SEQ EXT LSB field (called SEQ*), a (S)TS (EXT) LSB field (called TS*), an ID LSB field and a bit mask for additional fields. The M-bit is the RTP marker bit and the (S)TS (EXT) LSB is timestamp information sent with the least significant bits (the most significant bits are then expected to be unchanged compared to context). The timestamp information could either be the LSB of the (S)caled (T)ime(S)tamp value (if indicated with the t-bit unset) or the LSB of the absolute timestamp value. For profiles with a timestamp field in the compressed base header, the timestamp information is sent as an extended range to that field. The SEQ EXT LSB is extended range for the RTP sequence number. How extended range works is described in chapter 5.5.1 and 5.5.2. The t-bit is sent when timestamp is not scaled, otherwise it is always scaled with the timestamp delta. The ID LSB is the LSB of the IP Identification value. Various bit mask patterns are possible and can consist of S,H,C,D,T and I. The interpretations of these bits are given at the end of this chapter.

The guiding principle for choosing the extension type is to find the smallest header type that can communicate the information needed.

For the profiles defined in this document, four different extension sets are used, called A, B, C and D. Set A and C are for IPv6 and do not handle the IPv4 identification field, which set B and D do. Set A and B handle timestamp information which set C and D do not. All possible extensions are shown below with indications of which sets and types the extensions correspond to. For instance, B3 means that in extension set B, the extension is used with type value 3 (indicated in the type field).

The defined extension types are shown below:

```
                 0             7
           - - +-+-+-+-+-+-+-+-+
 A0, B0,       |0 0 0|   SEQ*  |
 C0, D0    - - +-+-+-+-+-+-+-+-+


                 0             7
```

```
                 - - +-+-+-+-+-+-+-+-+
    A1, B1           |0 0 1|M|  TS*  |
                 - - +-+-+-+-+-+-+-+-+
```

```
                                                1
                         0             7 8             5
                    - - +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       A2               |0 1 0|M|          TS*          |
                    - - +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

                                                1 1             2
                         0             7 8             5 6             3
                    - - +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       A3               |0 1 1|M|t|                 TS*                 |
                    - - +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+


                         0             7
                    - - +-+-+-+-+-+-+-+-+ - -
       A4               |1 0 0|M|H|D|T|t|
                    - - +-+-+-+-+-+-+-+-+ - -

                                                1
                         0             7 8             5
                    - - +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ - -
       A5               |1 0 1|M|C|H|S|D|      TS*      |
                    - - +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ - -

                                                1 1             2
                         0             7 8             5 6             3
                    - - +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ - -
       A6               |1 1 0|M|C|H|S|D|t|             TS*             |
                    - - +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ - -

                                                1
                         0             7 8             5
                    - - +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ - -
       A7               |1 1 1|M|C|H|S|D|T|t|    SEQ*   |
                    - - +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ - -

                                                1
                         0             7 8             5
                    - - +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       B2               |0 1 0|M|      TS*      |  SEQ* |
                    - - +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

                                                1
                         0             7 8             5
                    - - +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       B3               |0 1 1|M|  TS*  |     ID LSB    |
                    - - +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

```
                 0             7
             - - +-+-+-+-+-+-+-+-+
    B4, D4       |1 0 0|M| ID LSB|
             - - +-+-+-+-+-+-+-+-+


                 0             7
             - - +-+-+-+-+-+-+-+-+ - -
    B5           |1 0 1|M|H|D|T|I|
             - - +-+-+-+-+-+-+-+-+ - -

                                               1               2
                 0             7 8             5               3
             - - +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    B6           |1 1 0|M|t|         TS*         |     ID LSB    |
             - - +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

                                               1
                 0             7 8             5
             - - +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ - -
    B7           |1 1 1|M|C|H|S|D|T|I|t|   SEQ*  |
             - - +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ - -

                                               1
                 0             7 8             5
             - - +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    C1, D1       |0 0 1|M|         SEQ*          |
             - - +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

                                               1
                 0             7 8             5
             - - +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ - -
    C2, D2       |0 1 0|M|C|H|S|       SEQ*      |
             - - +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ - -


                 0             7
             - - +-+-+-+-+-+-+-+-+ - -
    C3, D3       |0 1 1|M|C|H|S|-|
             - - +-+-+-+-+-+-+-+-+ - -


                 0             7 8             5
             - - +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    D5           |1 0 1|M|        ID LSB         |
             - - +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

```
                                             1 1             2
                 0             7 8           5 6             3
             - - +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ - -
    D6           |1 1 0|M|C|H|S|-|           ID              |
             - - +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ - -
```

Bit masks indicating additional fields have the following meaning:

C - Traffic (C)lass / Type Of Service
H - (H)op Limit / Time To Live
S - Contributing (S)ources - CSRC
D - Timestamp (D)elta
T - (T)imestamp LSB
I - (I)dentification LSB

If any of these fields are included, they will appear in the order as listed above and the format of the fields will be as described below.

C - Traffic Class / Type Of Service

The field contains the value of the original IP header field.

```
         8 bits
- - +-+-+-+-+-+-+-+-+ - -
    |   TC / TOS    |
- - +-+-+-+-+-+-+-+-+ - -
```

H - Hop Limit / Time To Live

The field contains the value of the original IP header field.

```
         8 bits
- - +-+-+-+-+-+-+-+-+ - -
    |   HL / TTL    |
- - +-+-+-+-+-+-+-+-+ - -
```

S - Contributing Sources

The CSRC field is built up of:

- a counter of the number of CSRC items present (4 bits)
- an unused field (4 bits)
- the CSRC items, 1 to 15 (4-60 octets)

```
         1 octet    +     4 to 60 octets
- - +-+-+-+-+-+-+-+-+-+-+-+-+-+-+~~~+-+-+-+-+-+-+ - -
```

```
     | Count | Unused|      CSRC Items       |
   - - +-+-+-+-+-+-+-+-+-+-+-+-+-+-+~~~+-+-+-+-+-+-+ - -
```

D - Timestamp Delta

The Timestamp Delta field is a one-octet field. We want to communicate Timestamp Delta values corresponding to 80 ms. Therefore, the Timestamp Delta value communicated is not the actual number of samples, but the number of samples divided by 8. Thus, only Timestamp Delta values evenly divisible by 8 can be communicated in the Timestamp Delta field of an extension. On the other hand, the maximum value is 255*8 = 2040 (255 ms at 8000 Hz). Delta values larger than 2040 or delta values not evenly divisible by 8 must be communicated in a DYNAMIC packet.

```
          8 bits
- - +-+-+-+-+-+-+-+-+ - -
    |Timestamp Delta|
- - +-+-+-+-+-+-+-+-+ - -
```

Note that when the Timestamp Delta is changed, Timestamp LSB field MUST also be included not downscaled with the delta.

T - Timestamp LSB

The field contains the 16 least significant bits of the RTP timestamp, scaled if t-bit not set. May be sent as extended range for some profiles.

```
                 16 bits
- - +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ - -
    |              TS*              |
- - +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ - -
```

I - Identification

The field contains the IP Identification.

```
                 16 bits
- - +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |              ID               |
- - +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

When information of any kind is sent in an extension, the corresponding information SHOULD also be sent in some subsequent packets (either as Extensions or in DYNAMIC packets).

5.4.5. Feedback packets

Feedback packets are used by the decompressor to provide various types of feedback to the compressor. That could include active feedback to assure error free performance or passive feedback (in case of invalidated context) to request a context update from the compressor. The feedback mechanisms defined here leave a lot to the implementation regarding how to use feedback. The general feedback packet format is shown below:

```
                                  0   1   2   3   4   5   6   7
                                +...+...+...+...+...+...+...+...+
 FEEDBACK (GENERAL)             :   Context Identifier  (CID)   :
                                +---+---+---+---+---+---+---+---+
                                | 1   1   1   1   0 |    Type   |
                                +---+---+---+---+---+---+---+---+
```

Note that The CID field is present only for profiles using STATIC packet format 2 or 4, which are profiles supporting multiple packet streams. The Type field tells what kind of feedback the packet corresponds to and the feedback types defined are the following:

```
                                  0   1   2   3   4   5   6   7
                                +...+...+...+...+...+...+...+...+
 STATIC_FAILURE_FEEDBACK        :   Context Identifier  (CID)   :
                                +---+---+---+---+---+---+---+---+
                                | 1   1   1   1   0 | 0   0   0 |
                                +---+---+---+---+---+---+---+---+
```

The STATIC_FAILURE_FEEDBACK packet tells the compressor that the static part of the decompressor context is invalid, and that an update of that part is required. Reasons for sending such feedback could be that no STATIC packet has been received at all, or that decompression has failed even when DYNAMIC packets are decompressed.

```
                                  0   1   2   3   4   5   6   7
                                +...+...+...+...+...+...+...+...+
 INVALID_CONTEXT_FEEDBACK       :   Context Identifier  (CID)   :
                                +---+---+---+---+---+---+---+---+
                                | 1   1   1   1   0 | 0   0   1 |
                                +---+---+---+---+---+---+---+---+
                                |   Last Sequence Number LSB    |
                                +---+---+---+---+---+---+---+---+
```

The INVALID_CONTEXT_FEEDBACK packet SHOULD be sent to signal an invalid decompressor context, indicated by failing decompression of COMPRESSED packets.

```
                                   0   1   2   3   4   5   6   7
                                 +...+...+...+...+...+...+...+...+
    NO_PACKETS_FEEDBACK          :   Context Identifier  (CID)   :
                                 +---+---+---+---+---+---+---+---+
                                 | 1   1   1   1   0 | 0   1   0 |
                                 +---+---+---+---+---+---+---+---+
                                 |   Last Sequence Number LSB    |
                                 +---+---+---+---+---+---+---+---+
```

The NO_PACKET_FEEDBACK packet can be used by the decompressor to signal that packets have not been received for some time. It is not always possible for the decompressor to notice such events, and it is therefore up to the implementers to decide whether and when to use this feedback packet.

```
                                   0   1   2   3   4   5   6   7
                                 +...+...+...+...+...+...+...+...+
    LONGEST_LOSS_FEEDBACK        :   Context Identifier  (CID)   :
                                 +---+---+---+---+---+---+---+---+
                                 | 1   1   1   1   0 | 0   1   1 |
                                 +---+---+---+---+---+---+---+---+
                                 |   Last Sequence Number LSB    |
                                 +---+---+---+---+---+---+---+---+
                                 |    Length of longest loss     |
                                 +---+---+---+---+---+---+---+---+
```

The LONGEST_LOSS_FEEDBACK packet can be used by the decompressor to inform the compressor about the length of the longest loss event that has occurred on the link between compressor and decompressor. It is not always possible for the decompressor to provide this information, and it is therefore up to the implementers to decide whether and when to use this feedback packet.

```
                                   0   1   2   3   4   5   6   7
                                 +...+...+...+...+...+...+...+...+
    CONTEXT_UPDATED_FEEDBACK     :   Context Identifier  (CID)   :
                                 +---+---+---+---+---+---+---+---+
                                 | 1   1   1   1   0 | 1   0   0 |
                                 +---+---+---+---+---+---+---+---+
                                 |   Last Sequence Number LSB    |
                                 +---+---+---+---+---+---+---+---+
```

The CONTEXT_UPDATED_FEEDBACK packet can be used to signal that an update of some header fields has been correctly received, either in a DYNAMIC packet or in an EXTENDED_COMPRESSED packet. It is optional to use this active feedback mechanism and the compressor MUST NOT expect such packets initially. First after reception of one such packet, the compressor can expect to get this feedback from the decompressor.

5.5. Encoding of field values

The source increases the RTP sequence number by one for each packet sent. However, due to losses and reordering before the compression point, the changes seen by the compressor may vary. This would especially be the case if we consider the scenario in Figure 1.1 where there are cellular links at both ends of the path. That is one reason why sequence number changes need special treatment, but another reason is that both timestamps and IP identification for most packets can be recreated with a combination of history and sequence number knowledge. The profiles defined in this document handle the sequence number, timestamp and identification values with LSB encoding, except for some profiles that use LSP encoding for the sequence number. For timestamp, some profiles also use the principle with timer-based decompression. This chapter describes how the different encoding methods are applied to the various field values.

5.5.1. LSB encoding of field values

LSB encoding is used for sequence number, timestamp and identification encoding as described in chapter 5.3.1. The sequence numbers, included in all compressed headers, can be sent with extended range in extension headers. This is also the case with the timestamp value for profiles not using timer-based TS reconstruction (see 5.2.1 and 5.5.3). With timer-based timestamp decompression, the amount of timestamp LSB that is sent is always limited to the size of the field in the compressed header. Note that in most headers, the timestamp value is sent as STS LSB (scaled timestamp LSB), which means that it is the least significant bits of the timestamp, scaled down with the timestamp delta (STS LSB = LSB of [TS / TS Delta]).

5.5.2. LSP encoding of field values

LSP, as described in chapter 5.3.2, is used for sequence numbers in the "Sequence LSP" field of COMPRESSED1..COMPRESSED4 headers. For those headers, there are 28 code-points left for sequence information because 4 are reserved for packet type identification. An LSP of size 28 is therefore used with the following encoding:

```
CODE(n) = LSP:28(n)
```

The sequence range can be extended with extra bits in extension headers, as described in chapter 5.3.3. The "SEQ EXT LSB" field must for the case of extended LSP consist of the LSB of the integer quotient.

The reordering parameter for LSP MUST be set to 2 meaning that first and second order reordering can be handled by the encoding.

5.5.3. Timer-based timestamp decompression

The RTP timestamp field is one of the header fields that may change dynamically on a per packet basis. For audio services, the timestamp value can be inferred from the encoded RTP sequence number value during talk spurts. When the encoded sequence number is incremented by N, the timestamp value is incremented by N * Timestamp-Delta-value. However, when a talk spurt has faded into silence and a new talk spurt starts, the timestamp value will take a leap compared to the sequence number. To communicate this leap in the timestamp value, some additional action has to be taken. In chapter 5.4.4, extension headers are defined that can transfer this leap in the timestamp value. That increases, however, the average header size. This chapter describes an optional method used by some profiles (see the TbT column of table 5.1) to reconstruct the timestamp value, requiring only a fixed number of added bits for timestamp leaps. The method makes use of timers or a local wall clock at the decompressor.

To initialize the header compression and the timer-based timestamp reconstruction, the absolute value of the timestamp together with the sequence number must be transferred from compressor to decompressor at the beginning of the compression session. A default timestamp delta is also transferred. This is done through the transmission of a DYNAMIC header. For speech codecs with 8 kHz sampling frequency and 20 ms frame sizes, for example, the timestamp delta will be 8000*0.02 = 160. The decompressor then knows that the timestamp will increase by 160 for each packet containing 20 ms of speech. Hence, by using a local clock and by measuring packet arrival times, the decompressor can estimate the timestamp change compared to the previous packet. If, for example, a speech period has been succeeded by a silence period at the time T0 and a new speech period starts at the time T0+dT, it can be assumed that the timestamp has changed by:

```
round(dT/(time for one speech frame)) * (timestamp delta)
```

The packet time interval (or codec frame size in time) may be determined through the a priori knowledge that most speech codecs have constant frame sizes of 10, 20 or 30 ms, or through measurements on packet arrival times.

The decompressor can then get an estimate of the timestamp change, add this change to the previous value and replace the least significant bits with those received in the compressed header. This should give the correct timestamp value.

It is very important to verify the correctness of a timer-based timestamp decompression. However, this is automatically done in ROCCO with the header compression CRC verification.

5.6. Header compression CRCs, coverage and polynomials

This chapter contains a description of how to calculate the different CRCs used by the ROCCO profiles defined in this document.

5.6.1. STATIC packet CRC

The CRC in the STATIC header is calculated over the whole STATIC packet except for the header compression CRC itself. Therefore, the header compression CRC field MUST be set to 0 before the CRC calculation.

The CRC polynomial to be used in STATIC packets is:

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

5.6.2. DYNAMIC packet CRC

The CRC in the DYNAMIC packet is calculated over the original IP/UDP/RTP header. Before the calculation of the CRC, the IPv4 header checksum and the UDP checksum have to be set to 0. This makes it possible to recalculate the checksums after the decompression. Calculation over the full IP/UDP/RTP headers ensures that the decompressed IP/UDP/RTP header is a correct header.

The CRC polynomial to be used in DYNAMIC packets is:

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

5.6.3. COMPRESSED packet CRCs

COMPRESSED1..COMPRESSED4

The header compression CRC in COMPRESSED header types 1 to 4 is calculated over the same headers as the CRC in the DYNAMIC packet, except for profiles which use replacement of the UDP checksum, I.e. except for profiles 1-4 and 13-16. In profiles 1-4 and 13-16, the header compression CRC also covers the payload covered by the UDP checksum.

The polynomial to be used is:

C(x) = 1 + x + x^4 + x^5 + x^9 + x^10

COMPRESSED5..COMPRESSED8 and COMPRESSED14..COMPRESSED16

In COMPRESSED header types 5 to 8 and 14 to 16 the header compression CRC is calculated over the same headers as the CRC in the DYNAMIC packet, but with a different polynomial:

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

COMPRESSED10..COMPRESSED12

In COMPRESSED header types 10 to 12 the header compression CRC is calculated over the same headers as the CRC in the DYNAMIC packet, but with a different polynomial:

$$C(x) = 1 + x + x^3 + x^4 + x^6$$

6. Implementation issues

The profiles defined in this document specifies mechanisms for the protocol, while much of the usage of these mechanisms is left to the implementers to decide upon. This chapter is aimed to give guidelines, ideas and suggestions for implementing the scheme.

6.1. Compressor and decompressor logic, use of feedback

How to send and respond to the various kinds of FEEDBACK packets is not defined in this document, but left to the implementers to decide. However, it is recommended to reduce both the number of requests and the number of corresponding updating packets to a suitable level. Also it is recommended to use COMPRESSED packets with EXTENSIONS instead of DYNAMIC packets to update an invalid context, when possible.

More guidelines on this issue will be included here at a later date.

6.2. ROCCO over simplex links

This chapter contains a discussion about how ROCCO can be used over simplex links.

Previous chapters assumed that the decompressor has the possibility of sending requests to the compressor. This is true for many systems but there are several important scenarios where it is not possible to send information from the decompressor back to the compressor. The most important case may be when the packet are broadcasted in some way. It can also, for example, be the communication from a satellite.

Over a simplex link the decompressor does not have the possibility of sending information back to the compressor. The compressor does not know when the decompressor needs a STATIC or DYNAMIC packet. If STATIC and DYNAMIC packets are sent at regular intervals it is possible for the decompressor to recover a lost context. A slow-start mechanism and a periodic refresh guarantee that the decompressor can recover a lost synchronization fast.

The ROCCO scheme is especially suited for simplex links, since it is possible for the decompressor to continue with new reconstruction attempts, verified with the header checksum, until the next STATIC/DYNAMIC packet arrives. It is then possible for the decompressor to recover the context before the next STATIC/DYNAMIC packet arrives.

When ROCCO is used over simplex links it is RECOMMENDED that only one DYNAMIC packet be sent at a time and not several as stated in previous chapters.

6.2.1. Compression slow-start

When a field in the STATIC or DYNAMIC packet has changed or if we are at the beginning of a ROCCO session, it is necessary to send the STATIC/DYNAMIC packet to the decompressor. To ensure that the new information reaches the decompressor as fast as possible even if packets are lost over the link, a slow-start mechanism is used. After the first two packets (STATIC and DYNAMIC), STATIC and DYNAMIC packets (read refresh) are sent with an exponentially increasing period until a new change occurs. The following figure shows how the slow-start works:

```
|.|..|....|........|................|............................
^
Change in STATIC and/or DYNAMIC packet

Sent packets:
. Packet with compressed header
| STATIC packet followed by a DYNAMIC packet
```

6.2.2. Periodic refresh

To prevent the period between two refreshes from increasing too much, an upper limit on the interval between refreshes is set (MAX_PERIOD). This is used to avoid losing too many packets if the decompressor has lost its context. If the MAX_PERIOD between two refreshes is reached a new refresh (STATIC/DYNAMIC) has to be sent.

To avoid long time periods between two refreshes an upper limit on the time between two packets is used (MAX_TIME). This ensures that the time between two refreshes does not exceed the MAX_TIME even if no MAX_PERIOD packets are sent. If the MAX_TIME between two refreshes is reached, a new refresh (STATIC/DYNAMIC) has to be sent.

6.2.3. Refresh recommendations

It is recommended that the MAX_PERIOD not exceed 256 packets and that the maximum time between two refreshes (MAX_TIME) not exceed 5 seconds.

6.2.4. Cost and robustness of refreshes

If we assume that STATIC/DYNAMIC packets are sent every f'th packet, the average header size is:

```
  (S+U-C)
   ----- + C
     f
```

S = STATIC header size
U = DYNAMIC header size
C = COMPRESSED header size

The increase in average header size compared with the COMPRESSED header size is:

```
  (S+U-C)
   -----
     f
```

If we assume that we use ROCCO profile number 8 and that a refresh is sent every 256th packet, the increase in average header size is (18+15-2)/256=0.12 octets. The average header size for ROCCO profile number 4 over duplex links is, with realistic BER, 2.15 octets [PERF]. This results in an average header size of 2.15+0.12=2.27 octets.

The difference in robustness of ROCCO between simplex links and duplex links is very small. The reason for this is that the ROCCO decompressor very seldom loses its context. This results in that FEEDBACK packets are almost never needed, as proven in simulations. For example: In profile number 8 it is possible to lose up to 24 consecutive packets without losing the context in the decompressor. The probability that 25 consecutive packets are lost is very small even if channels with high bit error rates are used. This indicates that a ROCCO scheme implemented over simplex links is almost as robust as the duplex ROCCO scheme.

6.2.5. Simplex link improvements

DYNAMIC information can be sent in two different ways: either as an ordinary DYNAMIC packet or as extensions to COMPRESSED headers. If the information is sent in extensions to COMPRESSED headers, it is possible to reduce the average header size, since a COMPRESSED header with extension is smaller than a DYNAMIC header. If COMPRESSED headers are used for transmission of DYNAMIC information the following is important:

- All fields in the DYNAMIC packet that have changed since the last 3-4 refreshes have to be transmitted in the extension.
- DYNAMIC and STATIC packets still have to be sent at regular intervals to ensure that it is possible to recover a lost context even if COMPRESSED extension refreshes have failed.

6.3. Reverse decompression

This chapter describes an optional decompressor operation to reduce discarded packets due to an invalid context.

Once a context becomes invalid (e.g., in the case when more consecutive packet losses than expected has occurred), subsequent compressed packets cannot be decompressed correctly with normal decompression operation. This decompression operation aims at decompressing these packets with a later recovered context. The decompressor stores them until the context is validated. After the context is updated, the decompressor tries to recover the stored packets in the reverse order from the packet updating the context. Each time the stored packet is decompressed, its correctness is verified using the header compression CRC, which is transmitted in each compressed header. Correctly decompressed packets are transferred to upper layers in the original order.

Note that this reverse decompression introduces buffering while waiting for the context to be validated and thereby introduces additional delay. Thus, it should be used only when some amount of delay could be accepted. For example, for video packets belonging to the same video frame, the delay of packet arrival time does not cause presentation time delay. Delay-insensitive streaming applications can also be tolerant to such delay.

The following illustrates the decompression procedure in some detail:

1. The decompressor stores compressed packets that cannot be decompressed correctly due to an invalid context.

2. When the decompressor has received a context updating packet and

the context has been validated, it starts to recover the stored packets in reverse order. Decompression is carried out followed by the last decompressed packet to its previous packet as if the two packets were reordered. After that, decompressor checks the correctness of the reconstructed header using the header compression CRC.

3. If the header compression CRC indicates a successful decompression, the decompressor stores the complete packet and tries to decompress its previous packet. In this way, the stored packets are recovered from correctly decompressed packets until no compressed packets are left. For each packet, the decompressor checks the correctness of the decompressed headers using header compression CRC.

4. If the header compression CRC indicates an incorrectly decompressed packet, the reverse decompression attempt must be terminated and all remaining packets must be discarded.

5. Finally, the decompressor forwards all the correctly decompressed packets to upper layers in the original order.

6.4. Pre-verification of CRCs

For reasons of compression efficiency, it is desirable to keep the size of the header compression CRC as small as possible. However, if the size of the CRC is decreased, the reliability is also decreased and erroneous headers could be generated and passed on from the decompressor. It would then be desirable to find a method of increasing the strength of the CRC without making it larger.

There is one property of the ROCCO CRC and its usage that can be used to achieve this goal. The CRCs that will occur at the decompressor will in most cases follow a pattern well known also to the compressor. There are two factors that will affect which CRCs are generated and in which order they will occur. If the decompressor makes several reconstruction attempts, the first factor affecting the CRCs will be the order and properties of the assumptions made for each reconstruction attempt. The attempts are in general:

```
1:st attempt:    No loss is assumed
2:nd attempt:    Loss of the preceding packet is assumed
3:rd attempt:    Loss of the two preceding packets is assumed
4:th attempt:    Loss of the three preceding packets is assumed
etc.
```

The other factor that will affect the CRCs generated is what has really happened to preceding packets, that is, if no loss has occurred or if one or several preceding packets have been lost between compressor and decompressor.

Since the compressor knows how the decompressor performs the reconstruction attempts, the compressor can PRE-CALCULATE and VERIFY the most probable CRC situations. If a CRC is found not to detect an erroneous header, then a different packet type with a larger CRC (such as the "normal" COMPRESSED packet) should be used instead or additional information could be sent (by using EXTENDED_COMPRESSED or DYNAMIC packets). To ensure reliability, the important thing is that the CRC must fail if the header is not correctly reconstructed. Combining the two factors described above gives a list of the most probable CRCs that MUST fail.

- If ONE packet WAS lost, attempt one (no loss) MUST fail
- If TWO packets WERE lost, attempt one (no loss) MUST fail
- If TWO packets WERE lost, attempt two (one lost) MUST fail
- If THREE packets WERE lost, attempt one (no loss) MUST fail
- If THREE packets WERE lost, attempt two (one lost) MUST fail
- If THREE packets WERE lost, attempt three (two lost) MUST fail
- etc.

By doing PRE-CALCULATIONS of the six CRCs that would be the result of the events listed above, the CRC can be kept strong enough, even with a reduced size, because CRCs likely to fail will be avoided.

6.5. Using "guesses" with LSB and LSP encoding

ROCCO profiles using LSP encoding can handle 25 consecutive packet losses without invalidating the context. LSB or LSP encoding is also used for other fields and the range handled is then much larger. However, for all LSP or LSB decoding, the range can be extended with multiples by making reconstruction attempts (also called "guesses"). The limiting factors are implementation complexity and time. The following example shows how this can be done:

In chapter 5.3.2, LSP encoding is described. When an LSP encoded value for M code-points is decoded to a value S'(n), the original header can be reconstructed. If the CRC verification fails, a new reconstruction attempt could be made with S'(n)+M as the sequence number. If M was a multiple of 2 (LSB encoding), this would be the same as changing the value of the lowest MSB bit (i.e. the lowest bit NOT transmitted in LSB). More attempts could then be made increasing the sequence number by M for each attempt.

7. Further work

The ROCCO scheme, including the compression profiles optimized for conversational voice defined in this document, has been iterated and optimized for almost a year, and most of the desired functionality is today supported. However, much work remains before all details are settled. In addition to the tuning efforts, there are still new issues that should be investigated and implemented. This chapter elaborates on some ideas that might be sensible to apply to the scheme.

7.1. Compression of IPv6 extension headers

The ROCCO compression profiles defined in this document currently do not support compression of IPv6 extension headers, which is an undesirable limitation. Therefore, it is necessary to investigate what is really needed from the compression scheme regarding compression of extensions, and also to further develop the current and future compression profiles including the desired extension support.

7.2. Efficient compression of CSRC lists

The compression profiles defined in this document do support transmission of CSRC items, but this could probably be done in a much more efficient manner. Improved solutions for the CSRC compression would be preferable because if CSRC lists occur, the headers will be significantly expanded due to the size of the CSRC items.

7.3. General, media independent profiles

This document defines header compression profiles optimized for packet streams carrying conversational voice. Independently of packet stream characteristics, these profiles will successfully compress and decompress the headers of all IP/UDP/RTP packets. However, the compression may not be done in an optimal way. Therefore, general profiles should be designed that is optimized to handle uncharacterized or intermixed RTP packet streams as efficient as possible.

8. Implementation status

The ROCCO algorithm, as defined in previous versions of this Internet draft, has been implemented in a testbed environment for realtime IP traffic over wireless channels. In the testbed it is possible to listen to the effects of header compression in conjunction with packet losses. The currently implemented profiles are optimized for voice traffic only. A first rough estimate of the CPU utilization showed that ROCCO used only slightly more computational power than CRTP. On the other hand, with ROCCO the audio quality is significantly better. Figure 8.1 shows a block diagram of the testbed environment.

```
   +---------+   +---------------+   +----------+   +------------+
   | Speech  |-->| RTP/UDP/IP    |-->| Wired IP |-->| Header     |-->
   | Encoder |   | Encapsulation |   | Network  |   | Compressor |
   +---------+   +---------------+   +----------+   +------------+

            +----------+   +--------------+   +---------+
         ~~>| Cellular |~~>| Header De-   |-->| Speech  |
            | Link     |   | compressor   |   | Decoder |
            +----------+   +--------------+   +---------+
```

Figure 8.1 : Block diagram of testbed environment

The implementation has made some impact on the ROCCO protocol, realized in this document. Continuously updated information about implementation status can be obtained from the ROCCO homepage: http://www.ludd.luth.se/users/larsman/rocco/
See also [PERF] for simulated performance results.

9. Discussion and conclusions

This document has presented ROCCO, a robust header compression protocol framework adaptable to various usage and requirements. In addition to the general framework, realizations of the scheme optimized for conversational voice packet streams have been presented together with performance results for these realizations.

ROCCO uses CRCs to make local decompressor repairs of the context possible. Together with robust encoding methods for header fields, the usage of CRCs has made the scheme very robust and capable of coping with many consecutive packet losses (up to 25). One other important advantage with the CRC approach is that it makes the scheme reliable, meaning that it has a very low probability of incorrect header reconstruction and error propagation from the decompressor.

ROCCO defines a concept with header compression profiles, which is an abstraction that merges all scheme parameters into one. There are two fundamental advantages with the profile concept. First of all, only one scheme parameter must be negotiated by the link layer between the compression and the decompression points and this requirement does not change even if new internal header compression parameters are later added with new realizations of the scheme. The second advantage is the possibility of optimizing the scheme completely for all situations and functionality requirements by using different profiles.

Thanks to the profile concept, it has been possible to compress the headers down to a minimal size of 1 octet, the average header size being only 1.25 octets. As shown in [PERF], this is achieved without introducing any loss of packets due to invalid header compression context even over links with bit error rates as high as 1e-2. The probability of packet loss due to invalid header compression context is practically eliminated thanks to the robustness of ROCCO, even at the high BERs (1e-3 to 1e-2) a cellular system may produce.

With the profiles defined today in this document and in a separate document for conversational video [ROVID], ROCCO can efficiently compress IP/UDP/RTP packet streams for both conversational voice and video. There are profiles defined for both IPv4 and IPv6, support for various numbers of concurrent packet streams, enabled or disabled UDP checksum, etc. Optimizations have been done to efficiently take care of the IPv4 Identification field and make it more compressible if knowledge about the sender's assignment policy can be obtained. Finally, it is also possible to tune the compression scheme to the characteristics of the channel it is used over.

ROCCO has been evolved and improved during one year. Experiences from implementations have been taken into account in the process of improving the scheme. However, even if many suggestions for efficient implementations are included, there is still room for implementers to find even more efficient realizations.

Hence, ROCCO provides at the present date a powerful toolbox for achieving efficient and robust header compression in various types of scenarios and over various types of links. ROCCO has also been proven to be suitable for cellular environments.

10. Security considerations

Because encryption eliminates the redundancy that header compression schemes try to exploit, there is some inducement to forego encryption in order to achieve operation over low-bandwidth links. However, for those cases where encryption of data (and not headers) is sufficient, RTP does specify an alternative encryption method in which only the RTP payload is encrypted and the headers are left in the clear. That would still allow header compression to be applied.

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 IP, UDP and RTP headers and possibly even valid UDP checksums. Such corruption may be detected with end-to-end authentication and integrity mechanisms which will not be affected by the compression. Further, this header compression scheme provides an internal checksum for verification of re-constructed headers. This 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 STATIC, DYNAMIC 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.

11. Acknowledgements

When designing this protocol, earlier header compression ideas described in [CJHC], [IPHC] and [CRTP] have been important sources of knowledge.

Thanks to Takeshi Yoshimura at NTT DoCoMo for providing the reverse decompression chapter (chapter 6.3). Thanks also to Anton Martensson for many valuable draft contributions and to Andreas Jonsson (Lulea University), who made a great job supporting this work in his study of header field change patterns. Thanks also to all others who have given comments.

12. Intellectual property considerations

This proposal in is conformity with RFC 2026.

Telefonaktiebolaget LM Ericsson and its subsidiaries, in accordance with corporate policy, will for submissions rightfully made by its employees which are adopted or recommended as a standard by the IETF offer patent licensing as follows:

If part(s) of a submission by Ericsson employees is (are) included in a standard and Ericsson has patents and/or patent application(s) that are essential to implementation of such included part(s) in said standard, Ericsson is prepared to grant - on the basis of reciprocity (grant-back) - a license on such included part(s) on reasonable, non-discriminatory terms and conditions.

For the avoidance of doubt this general patent licensing undertaking applies to this proposal.

13. References

[UDP] Jon Postel, "User Datagram Protocol", RFC 768, August 1980.

[IPv4] Jon Postel, "Internet Protocol", RFC 791, September 1981.

[IPv6] Steven Deering, Robert Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

[RTP] Henning Schulzrinne, Stephen Casner, Ron Frederick, Van Jacobson, "RTP: A Transport Protocol for Real-Time Applications", RFC 1889, January 1996.

[HDLC] William Simpson, "PPP in HDLC-like framing", RFC 1662, 1994.

[VJHC] Van Jacobson, "Compressing TCP/IP Headers for Low-Speed Serial Links", RFC 1144, February 1990.

[IPHC] Mikael Degermark, Bjorn Nordgren, Stephen Pink, "IP Header Compression", RFC 2507, February 1999.

[CRTP] Steven Casner, Van Jacobson, "Compressing IP/UDP/RTP Headers for Low-Speed Serial Links", RFC 2508, February 1999.

[PPPHC] Mathias Engan, Steven Casner, Carsten Bormann, "IP Header Compression over PPP", RFC 2509, February 1999.

[CRTPC] Mikael Degermark, Hans Hannu, Lars-Erik Jonsson, Krister Svanbro, "CRTP over cellular radio links", Internet Draft (work in progress), December 1999. <draft-degermark-crtp-cellular-01.txt>

[CELL] Lars Westberg, Morgan Lindqvist, "Realtime traffic over cellular access networks", Internet Draft (work in progress), May 2000. <draft-westberg-realtime-cellular-02.txt>

[ROVID] Anton Martensson, Torbjorn Einarsson, Lars-Erik Jonsson, "ROCCO Conversational Video Profiles", Internet Draft (work in progress), May 2000. <draft-ietf-rohc-rtp-rocco-video.txt>

[PERF] Hans Hannu, Krister Svanbro, Lars-Erik Jonsson, "ROCCO Performance Evaluation", Internet Draft (work in progress), May 2000. <draft-ietf-rohc-rtp-rocco-performance-00.txt,ps>

[LLG] Krister Svanbro, "Lower Layer Guidelines for Robust Header Compression", Internet Draft (work in progress), May 2000.

<draft-ietf-rohc-lower-layer-guidelines-00.txt>

[WCDMA] "Universal Mobile Telecommunications System (UMTS); Selection procedures for the choice of radio transmission technologies of the UMTS (UMTS 30.03 version 3.1.0)". ETSI TR 101 112 V3.0.1, November 1997.

14. Authors' addresses

Lars-Erik Jonsson	Tel: +46 920 20 21 07
Ericsson Erisoft AB	Fax: +46 920 20 20 99
Box 920	Mobile: +46 70 554 82 71
SE-971 28 Lulea, Sweden	EMail: lars-erik.jonsson@ericsson.com

Mikael Degermark	Tel: +46 920 911 88
Dept of CS & EE	Fax: +46 920 728 01
Lulea University of Technology	Moblie: +46 70 833 89 33
SE-971 87 Lulea	EMail: micke@sm.luth.se

Hans Hannu	Tel: +46 920 20 21 84
Ericsson Erisoft AB	Fax: +46 920 20 20 99
Box 920	Mobile: +46 70 559 90 15
SE-971 28 Lulea, Sweden	EMail: hans.hannu@ericsson.com

Krister Svanbro	Tel: +46 920 20 20 77
Ericsson Erisoft AB	Fax: +46 920 20 20 99
Box 920	Mobile: +46 70 531 25 08
SE-971 28 Lulea, Sweden	EMail: krister.svanbro@ericsson.com

Appendix A. Detailed classification of header fields

Header compression is possible due to the fact that most header fields do not vary randomly from packet to packet. Many of the fields exhibit static behavior or changes in a more or less predictable way. When designing a header compression scheme, it is of fundamental importance to understand the behavior of the fields in detail.

In this appendix, all IP, UDP and RTP header fields are classified and analyzed in two steps. First, we have a general classification in A.1 where the fields are classified based on stable knowledge and assumptions. The general classification does not take into account the change characteristics of changing fields because those will vary more or less depending on the implementation and on the application used. A less stable but more detailed analysis considering the change characteristics is then done in A.2. Finally, A.3 summarizes this appendix with conclusions about how the various header fields should be handled by the header compression scheme to optimize compression and functionality.

A.1. General classification

On a general level, the header fields are separated into 5 classes:

INFERRED These fields contain values that can be inferred from other values, for example the size of the frame carrying the packet, and thus does not have to be handled at all by the compression scheme.

STATIC These fields are expected to be constant throughout the lifetime of the packet stream. Static information must in some way be communicated once.

STATIC-DEF STATIC fields whose values define a packet stream. They are in general handled as STATIC.

STATIC-KNOWN These STATIC fields are expected to have well-known values and therefore do not need to be communicated at all.

CHANGING These fields are expected to vary in some way, either randomly, within a limited value set or range, or in some other manner.

In this section, each of the IP, UDP and RTP header fields is assigned to one of these classes. For all fields except those classified as CHANGING, the motives for the classification are also stated. CHANGING fields are in A.2 further examined and classified based on their expected change behavior.

A.1.1. IPv6 header fields

Field	Size (bits)	Class
Version	4	STATIC-KNOWN
Traffic Class	8	CHANGING
Flow Label	20	STATIC-DEF
Payload Length	16	INFERRED
Next Header	8	STATIC-KNOWN
Hop Limit	8	CHANGING
Source Address	128	STATIC-DEF
Destination Address	128	STATIC-DEF

Version

The version field states which IP version the packet is based on. Packets with different values in this field must be handled by different IP stacks. For header compression, different compression profiles must also be used. When compressor and decompressor have negotiated which profile to use, the IP version is also known to both parties. The field is therefore classified as STATIC-KNOWN.

Flow Label

This field may be used to identify packets belonging to a specific packet stream. If not used, the value should be set to zero. Otherwise, all packets belonging to the same stream must have the same value in this field, it being one of the fields defining the stream. The field is therefore classified as STATIC-DEF.

Payload Length

Information about the packet length (and then also payload length) is expected to be provided by the link layer. The field is therefore classified as INFERRED.

Next Header

This field is expected to have the same value in all packets of a packet stream. As for the version number, a certain compression profile can only handle a specific next header which means that this value is known when profile has been negotiated. The field is therefore classified as STATIC-KNOWN.

Source and Destination addresses

These fields are part of the definition of a stream and must thus be constant for all packets in the stream. The fields are therefore classified as STATIC-DEF.

Summarizing the bits corresponding to the classes gives:

Class	Size (octets)
INFERRED	2
STATIC-DEF	34.5
STATIC-KNOWN	1.5
CHANGING	2

A.1.2. IPv4 header fields

Field	Size (bits)	Class
Version	4	STATIC-KNOWN
Header Length	4	STATIC-KNOWN
Type Of Service	8	CHANGING
Packet Length	16	INFERRED
Identification	16	CHANGING
Reserved flag	1	STATIC-KNOWN
May Fragment flag	1	STATIC
Last Fragment flag	1	STATIC-KNOWN
Fragment Offset	13	STATIC-KNOWN
Time To Live	8	CHANGING
Protocol	8	STATIC-KNOWN
Header Checksum	16	INFERRED
Source Address	32	STATIC-DEF
Destination Address	32	STATIC-DEF

Version

The version field states which IP version the packet is based on and packets with different values in this field must be handled by different IP stacks. For header compression, different compression profiles must also be used. When compressor and decompressor has negotiated which profile to use, the IP version is also well known to both parties. The field is therefore classified as STATIC-KNOWN.

Header Length

As long as there are no options present in the IP header, the header length is constant and well known. If there are options, the fields would be STATIC, but we assume no options. The field is therefore classified as STATIC-KNOWN.

Packet Length

Information about the packet length is expected to be provided by the link layer. The field is therefore classified as INFERRED.

Flags

The Reserved flag must be set to zero and is therefore classified as STATIC-KNOWN. The May Fragment flag will be constant for all packets in a stream and is therefore classified as STATIC. Finally, the Last Fragment bit is expected to be zero because fragmentation is NOT expected, due to the small packet size expected. The Last Fragment bit is therefore classified as STATIC-KNOWN.

Fragment Offset

With the assumption that no fragmentation occurs, the fragment offset is always zero. The field is therefore classified as STATIC-KNOWN.

Protocol

This field is expected to have the same value in all packets of a packet stream. As for the version number, a certain compression profile can only handle a specific next header which means that this value is well known when profile has been negotiated. The field is therefore classified as STATIC-KNOWN.

Header Checksum

The header checksum protects individual hops from processing a corrupted header. When almost all IP header information is compressed away, there is no need to have this additional checksum; instead it can be regenerate at the decompressor side. The field is therefore classified as INFERRED.

Source and Destination addresses

These fields are part of the definition of a stream and must thus be constant for all packets in the stream. The fields are therefore classified as STATIC-DEF.

Summarizing the bits corresponding to the classes gives:

Class	Size (octets)
INFERRED	4
STATIC	1 bit
STATIC-DEF	8
STATIC-KNOWN	3 +7 bits
CHANGING	4

A.1.3. UDP header fields

Field	Size (bits)	Class
Source Port	16	STATIC-DEF
Destination Port	16	STATIC-DEF
Length	16	INFERRED
Checksum	16	CHANGING

Source and Destination ports

These fields are part of the definition of a stream and must thus be constant for all packets in the stream. The fields are therefore classified as STATIC-DEF.

Length

This field is redundant and is therefore classified as INFERRED.

Summarizing the bits corresponding to the classes gives:

Class	Size (octets)
INFERRED	2
STATIC-DEF	4
CHANGING	2

A.1.4. RTP header fields

Field	Size (bits)	Class
Version	2	STATIC-KNOWN
Padding	1	STATIC
Extension	1	STATIC
CSRC Counter	4	CHANGING
Marker	1	CHANGING
Payload Type	7	CHANGING
Sequence Number	16	CHANGING
Timestamp	32	CHANGING
SSRC	32	STATIC-DEF
CSRC	0(-480)	CHANGING

Version

There exists only one working RTP version and that is version 2. The field is therefore classified as STATIC-KNOWN.

Padding

The use of this field depends on the application, but when payload padding is used it is likely to be present in all packets. The field is therefore classified as STATIC.

Extension

If RTP extensions is used by the application, it is likely to be an extension present in all packets (but use of extensions is very uncommon). However, for safety's sake this field is classified as STATIC and not STATIC-KNOWN.

SSRC

This field is part of the definition of a stream and must thus be constant for all packets in the stream. The field is therefore classified as STATIC-DEF.

Summarizing the bits corresponding to the classes gives:

Class	Size (octets)
STATIC	2 bits
STATIC-DEF	4
STATIC-KNOWN	2 bits
CHANGING	7.5(-67.5)

A.1.5. Summary for IP/UDP/RTP

If we summarize this for IP/UDP/RTP we get:

Class \ IP ver	IPv6 (octets)	IPv4 (octets)
INFERRED	4	6
STATIC	2 bits	3 bits
STATIC-DEF	42.5	16
STATIC-KNOWN	1 +6 bits	4 +1 bit
CHANGING	11.5(-71.5)	13.5(-73.5)
Total	60(-120)	40(-100)

A.2. Analysis of change patterns of header fields

To design suitable mechanisms for efficient compression of all header fields, their change patterns must be analyzed. For this reason, an extended classification is done based on the general classification in A.1, considering the fields which were labeled CHANGING in that classification. Different applications will use the fields in different ways, which may affect their behavior. When this is the case, typical behavior for conversational audio and video will be discussed.

The CHANGING fields are separated into five different subclasses:

STATIC	These are fields that were classified as CHANGING on a general basis, but are classified as STATIC here due to certain additional assumptions.
SEMISTATIC	These fields are STATIC most of the time. However, occasionally the value changes but reverts to its original value after a known number of packets.
RARELY-CHANGING (RC)	These are fields that change their values occasionally and then keep their new values.
ALTERNATING	These fields alternate between a small number of different values.
IRREGULAR	These, finally, are the fields for which no useful change pattern can be identified.

To further expand the classification possibilities without increasing complexity, the classification can be done either according to the values of the field and/or according to the values of the deltas for the field.

When the classification is done, other details are also stated regarding possible additional knowledge about the field values and/or field deltas, according to the classification. For fields classified as STATIC or SEMISTATIC, the case could be that the value of the field is not only STATIC but also well KNOWN a priori (two states for SEMISTATIC fields). For fields with non-irregular change behavior, it could be known that changes usually are within a LIMITED range compared to the maximal change for the field. For other fields, the values are completely UNKNOWN.

Table A.1 classifies all the CHANGING fields based on their expected change patterns, especially for conversational audio and video.

Field		Value/Delta	Class	Knowledge
IPv4 Id:	Sequential	Delta	STATIC	KNOWN
	Seq. jump	Delta	RC	LIMITED
	Random	Value	IRREGULAR	UNKNOWN
IP TOS / Tr. Class		Value	RC	UNKNOWN
IP TTL / Hop Limit		Value	ALTERNATING	LIMITED
UDP Checksum:	Disabled	Value	STATIC	KNOWN
	Enabled	Value	IRREGULAR	UNKNOWN
RTP CSRC Count:	No mix	Value	STATIC	KNOWN
	Mixed	Value	RC	LIMITED
RTP Marker		Value	SEMISTATIC	KNOWN/KNOWN
RTP Payload Type		Value	RC	UNKNOWN
RTP Sequence Number		Delta	STATIC	KNOWN
RTP Timestamp		Delta	RC	LIMITED
RTP CSRC List:	No mix	-	-	-
	Mixed	Value	RC	UNKNOWN

Table A.1 : Classification of CHANGING header fields

The following subsections discuss the various header fields in detail. Note that table A.1 and the discussions below do not consider changes caused by loss or reordering before the compression point.

A.2.1. IPv4 Identification

The Identification field (IP ID) of the IPv4 header is there to identify which fragments constitute a datagram when reassembling fragmented datagrams. The IPv4 specification does not specify exactly how this field is to be assigned values, only that each packet should get an IP ID that is unique for the source-destination pair and protocol for the time the datagram (or any of its fragments) could be alive in the network. This means that assignment of IP ID values can be done in various ways, which we have separated into three classes.

Sequential

This assignment policy keeps a separate counter for each outgoing packet stream and thus the IP ID value will increment by one for each packet in the stream. Therefore, the delta value of the field is constant and well known a priori. When RTP is used on top of UDP and IP, the IP ID value follows the RTP sequence number. This assignment policy is the most desirable for header compression purposes but its usage is not as common as it should be. The reason is that it can be realized only if UDP and IP are implemented together so that UDP, which separates packet streams by the port identification, can make IP use separate ID counters for each packet stream.

Sequential jump

This is the most common assignment policy in today's IP stacks. The difference from the sequential method is that only one counter is used for all connections. When the sender is running more than one packet stream simultaneously, the IP ID can increase by more than one. The IP ID values will be much more predictable and require less bits to transfer than random values, and the packet-to-packet increment (determined by the number of active outgoing packet streams and sending frequencies) will usually be limited.

Random

Some IP stacks assign IP ID values using a pseudo-random number generator. There is thus no correlation between the ID values of subsequent datagrams. Therefore there is no way to predict the IP ID value for the next datagram. For header compression purposes, this means that the IP ID field needs to be sent uncompressed with each datagram, resulting in two extra octets of header. IP stacks in cellular terminals SHOULD NOT use this IP ID assignment policy.

It should be noted that the ID is an IPv4 mechanism and is therefore not needed at all in IPv6 profiles. For IPv4 the ID could be handled in three different ways. Firstly, we have the inefficient but reliable solution where the ID field is sent as-is in all packets, increasing the compressed headers with two octets. This is the best way to handle the ID field if the sender uses random assignment of the ID field. Secondly, there can be solutions with more flexible mechanisms requiring less bits for the ID handling as long as sequential jump assignment is used. Such solutions will probably require even more bits if random assignment is used by the sender. Knowledge about the sender's assignment policy could therefore be useful when choosing between the two solutions above. Finally, even for IPv4, header compression could be designed without any additional information for the ID field included in compressed headers. To use such schemes, it must be known that the sender makes use of the pure sequential assignment policy for the ID field. That might not be possible to know, which implies that the applicability of such solutions is very uncertain. However, designers of IPv4 stacks for cellular terminals SHOULD use the sequential policy.

A.2.2. IP Traffic-Class / Type-Of-Service

The Traffic-Class (IPv6) or Type-Of-Service (IPv4) field is expected to be constant during the lifetime of a packet stream or to change relatively seldom.

A.2.3. IP Hop-Limit / Time-To-Live

The Hop-Limit (IPv6) or Time-To-Live (IPv4) field is expected to be constant during the lifetime of a packet stream or to alternate between a limited number of values due to route changes.

A.2.4. UDP Checksum

The UDP checksum is optional. If disabled, its value is constantly zero and could be compressed away. If enabled, its value depends on the payload, which for compression purposes is equivalent to it changing randomly with every packet.

A.2.5. RTP CSRC Counter

This is a counter indicating the number of CSRC items present in the CSRC list. This number is expected to be almost constant on a packet-to-packet basis and change by small amount. As long as no RTP mixer is used, the value of this field is zero.

A.2.6. RTP Marker

For audio the marker bit should be set only in the first packet of a talkspurt while for video it should be set in the last packet of every picture. This means that in both cases the RTP marker is classified as SEMISTATIC with well-known values for both states.

A.2.7. RTP Payload Type

Changes of the RTP payload type within a packet stream are expected to be rare. Applications could adapt to congestion by changing payload type and/or frame sizes, but that is not expected to happen frequently.

A.2.8. RTP Sequence Number

The RTP sequence number will be incremented by one for each packet sent.

A.2.9. RTP Timestamp

In the audio case:

As long as there are no pauses in the audio stream, the RTP timestamp will be incremented by a constant delta, corresponding to the number of samples in the speech frame. It will thus mostly follow the RTP sequence number. When there has been a silent period and a new talkspurt begins, the timestamp will jump in proportion to the length of the silent period. However, the increment will probably be within a relatively limited range.

In the video case:

The timestamp change between two consecutive packets will either be zero or increase by a multiple of a fixed value corresponding to the picture clock frequency. The timestamp can also decrease by a multiple of the fixed value if B-pictures are used. The delta interval, expressed as a multiple of the picture clock frequency, is in most cases very limited.

A.2.10. RTP Contributing Sources (CSRC)

The participants in a session, which are identified by the CSRC fields, are expected to be almost the same on a packet-to-packet basis with relatively few additions or removals. As long as RTP mixers are not used, no CSRC fields are present at all.

A.3. Header compression strategies

This section elaborates on what has been done in previous sections. Based in the classifications, recommendations are given on how to handle the various fields in the header compression process. Seven different actions are possible and these are listed together with the fields to which each action applies.

A.3.1. Do not send at all

The fields that have well known values a priori do not have to be sent at all. These are:

- IP Version
- IPv6 Payload Length
- IPv6 Next Header
- IPv4 Header Length
- IPv4 Reserved Flag
- IPv4 Last Fragment Flag
- IPv4 Fragment Offset
- IPv4 Protocol
- UDP Checksum (if disabled)
- RTP Version

A.3.2. Transmit only initially

The fields that are constant throughout the lifetime of the packet stream have to be transmitted and correctly delivered to the decompressor only once. These are:

- IP Source Address
- IP Destination Address
- IPv6 Flow Label
- IPv4 May Fragment Flag
- UDP Source Port
- UDP Destination Port
- RTP Padding Flag
- RTP Extension Flag
- RTP SSRC

A.3.3. Transmit initially, but be prepared to update occasionally

The fields that are changing only occasionally must be transmitted initially but there must also be a way to update these fields with new values if they change. These fields are:

- IPv6 Traffic Class
- IPv6 Hop Limit

- IPv4 Type Of Service (TOS)
- IPv4 Time To Live (TTL)
- RTP CSRC Counter
- RTP Payload Type
- RTP CSRC List

A.3.4. Be prepared to update or send as-is frequently

For fields that normally are either constant or whose values can be deduced from some other field but frequently diverge from that behavior, there must be an efficient way to update the field value or send it as-is in some packets. Those fields are:

- IPv4 Identification (if not sequentially assigned)
- RTP Marker
- RTP Timestamp

A.3.5. Guarantee continuous robustness

Fields that behave like a counter with a fixed delta for ALL packets, the only requirement on the transmission encoding is that packet losses between compressor and decompressor must be tolerable. If more than one such field exists, all these can be communicated together. Such fields can also be used to interpret the values for fields listed in the previous section. Fields that have this counter behavior are:

- IPv4 Identification (if sequentially assigned)
- RTP Sequence Number

A.3.6. Transmit as-is in all packets

Fields that have completely random values for each packet must be included as-is in all compressed headers. Those fields are:

- IPv4 Identification (if randomly assigned)
- UDP Checksum (if enabled)

A.3.7. Establish and be prepared to update delta

Finally, there is a field that is usually increasing by a fixed delta and is correlated to another field. For this field it would make sense to make that delta part of the context state. The delta must then be possible to initiate and update in the same way as the fields listed in A.3.3. The field to which this applies is:

- RTP Timestamp

This Internet-Draft expires December 15, 2000.