Network Working Group
INTERNET-DRAFT
Expires: September 2000

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

March 10, 2000

RObust Checksum-based header COmpression (ROCCO)
<draft-jonsson-robust-hc-04.ps>

Status of this memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or cite them other than as "work in progress".

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/lid-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html

This document is an individual submission to the IETF. Comments should be directed to the authors.

Abstract

IP/UDP/RTP header compression [CRTP] can generate a large number of lost packets when used over links with significant error rates, especially when the round-trip time of the link is long.

This document describes a more robust header compression scheme. The scheme is adaptable to the characteristics of the link over which it is used and also to the properties of the packet streams it compresses. Robustness against link loss is achieved without decreasing compression efficiency.

Table of contents

1. Introduction.....5
2. Terminology.....6
3. Existing header compression schemes.....9
4. Desired improvements.....11
5. Proposed solution.....11
5.1. Header compression framework.....12
5.2. General ROCCO principles.....12
5.3. Data structures.....13
5.4. Header compression profiles.....13
5.5. Profile negotiation.....14
5.6. Link layer requirements.....15
6. Classification of header fields.....15
7. Header compression profiles for IP telephony packet streams..16
7.1. Usage scenarios, environment and requirements.....16
7.2. Analysis of change patterns of header fields.....17
7.2.1. IPv4 Identification.....19
7.2.2. IP Traffic-Class / Type-Of-Service.....20
7.2.3. IP Hop-Limit / Time-To-Live.....20
7.2.4. UDP Checksum.....20
7.2.5. RTP CSRC Counter.....20
7.2.6. RTP Marker.....21
7.2.7. RTP Payload Type.....21
7.2.8. RTP Sequence Number.....21
7.2.9. RTP Timestamp.....21
7.2.10. RTP Contributing Sources (CSRC).....21
7.3. Profile definitions.....21
7.3.1. List of defined profiles.....21
7.3.2. Additional common profile characteristics....24
7.4. Encoding methods used.....24
7.4.1. Least Significant Bits (LSB) encoding.....25
7.4.2. Least Significant Part (LSP) encoding.....25
7.5. Packet formats.....26
7.5.1. Static information packets, initialization...27
7.5.2. Dynamic information packets.....28
7.5.3. Compressed packets.....31
7.5.4. Minimal compressed headers.....31
7.5.5. Extensions to compressed headers.....32
7.5.6. Feedback packets.....37
7.6. Interpretations of the code field.....39
7.7. Encoding of field values.....40
7.7.1. LSP encoding of field values.....40
7.7.2. LSB encoding of field values.....40
7.7.3. Sequence encoding with no information.....41
7.8. Header compression CRCs, coverage and polynomials.....41
7.8.1. STATIC packet CRC.....41
7.8.2. DYNAMIC packet CRC.....41
7.8.3. COMPRESSED packet CRCs.....42

8. Implementation issues.....42
8.1. Feedback and context update procedures.....42
8.2. ROCCO over simplex links.....42
8.2.1. Compression slow-start.....43
8.2.2. Periodic refresh.....43
8.2.3. Refresh recommendations.....44
8.2.4. Cost and robustness of refreshes.....44
8.2.5. Simplex link improvements.....45
8.3. Pre-verification of CRCs.....45
8.4. Using "guesses" with LSB and LSP encoding.....46
9. Further work.....47
9.1. Timer-based timestamp reconstruction.....47
9.2. Compression of IPv6 extension headers.....48
9.3. Replacement of the UDP checksum.....49
9.4. Efficient compression of CSRC lists.....49
9.5. General, media independent profiles.....49
10. Implementation status.....50
11. Discussion and conclusions.....50
12. Security considerations.....51
13. Acknowledgements.....52
14. Intellectual property considerations.....52
15. References.....53
16. Authors' addresses.....54

Appendix A. Detailed classification of header fields.....55
A.1. IPv6 header fields.....55
A.2. IPv4 header fields.....56
A.3. UDP header fields.....58
A.4. RTP header fields.....59
A.5. Summary.....60

Appendix B. Simulated performance results.....61
B.1. Simulated scenario.....61
B.2. Input data.....61
B.3. Influence of pre-HC links.....61
B.4. Used link layers.....62
B.5. The cellular link.....62
B.6. Compression performance.....62
B.7. Robustness results.....64
B.8. CRC strength considerations.....66

Document history

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. New simulation results are included in an appendix.

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.

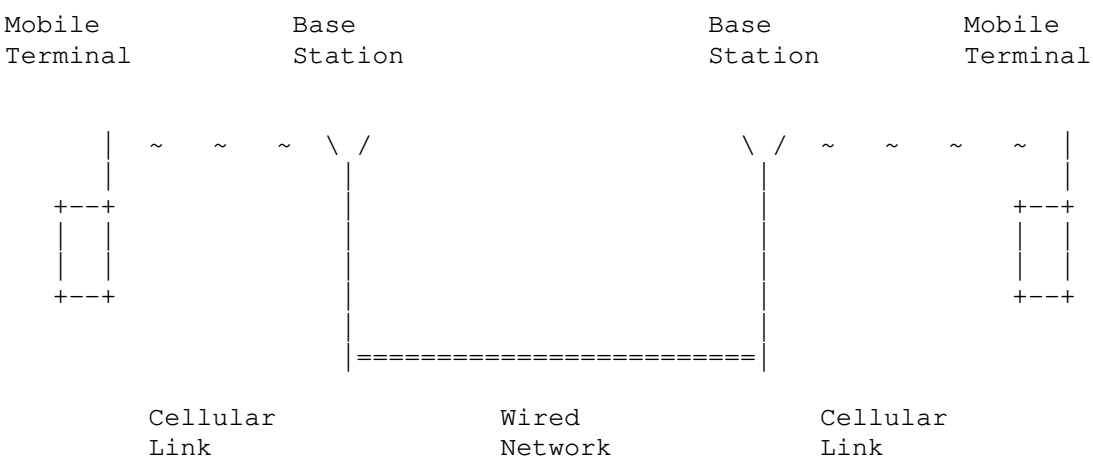


Figure 1.1 : Scenario for IP telephony over cellular links

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 two parameters, compression efficiency and robustness. The compression efficiency is determined by how much header sizes are reduced by the compression scheme.

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 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 two parameters, compression efficiency and robustness. 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.

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. 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.

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.

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.

4. Desired improvements

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.

5. Proposed solution

We propose a solution which is heavily geared towards local repair of the context. 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 our 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.

5.1. Header compression framework

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.

5.2. 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.

5.3. 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.

5.4. 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, but for each logical channel exactly one profile is active. 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 compression 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 streams supported.
- 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 7 defines compression profiles for IP telephony to use over cellular radio links.

5.5. 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.

5.6. Link layer requirements

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

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.

6. Classification of header fields

The IP/UDP/RTP header fields can be classified according to the way they are expected to change. On a general level, we classify them as:

INFERRED	These fields contain values that can be inferred from other values, for example the size of the frame carrying the packet, and thus must not be handled at all by the compression scheme.
STATIC	These fields are expected to be constant during 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.

All unchanging fields of the IP/UDP/RTP headers are classified in Appendix A. Table 6.1 summarizes this for IP/UDP/RTP. The interval for the CHANGING fields in Table 6.1 reflects the varying size of the CSRC list of the RTP header.

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)

Table 6.1 : Sizes of field classes

The information carried by the STATIC and STATIC-DEF fields has to be transferred once, and every header compression profile MUST specify a way of doing this. The information in INFERRED and STATIC-KNOWN fields SHOULD NOT be transmitted at all. The values of INFERRED fields can be computed from other information known to the decompressor. The values of STATIC-KNOWN fields are implicitly defined by the compression profiles. Profiles MUST also handle the CHANGING fields, and that should be done efficiently based on the expected change patterns for the kind of packet streams the profile compresses. A detailed analysis of the change patterns of these fields SHOULD therefore also be done for each profile.

7. Header compression profiles for IP telephony packet streams

This section exemplifies how the framework outlined in chapter 5 could be instantiated by defining profiles for header compression of IP telephony packet streams. A number of profiles are defined providing support for both IPv6 and IPv4 in combination with various functionality.

7.1. Usage scenarios, environment and requirements

These profiles are intended for IP telephony over cellular links with high error rates. 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 a cellular 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.

7.2. Analysis of change patterns of header fields

To design suitable mechanisms for efficient compression of all header fields, their change patterns need to be analyzed. For this reason, an extended classification tailored for IP-telephony packet streams is made, which applies to all profiles defined in this document. This classification is based on the general classification in chapter 6 and Appendix A, and considers the fields which were labeled CHANGING in that classification. These fields are further separated into five different subclasses:

STATIC	These are fields that were classified as CHANGING on a general basis, but are classified as STATIC for IP telephony packet streams.
SEMISTATIC	These fields are STATIC most of the time. However, occasionally the value changes but returns 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 few 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 on the values of the field and/or on the values of the deltas for the field.

When the classification is done, other details could also be 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 fileds, the values are completely UNKNOWN.

Table 7.1 classifies all the CHANGING fields based on their expected change patterns for IP telephony streams.

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 7.1 : Classification of CHANGING header fields

The following subsections discuss the various header fields in detail. Note that table 7.1 and the discussions below do not consider changes caused by loss or reordering before the compression point.

7.2.1. 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) 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.

In this document, various profiles are defined with different IP ID capabilities. First of all, it should be noted that the ID is an IPv4 mechanism and is therefore not needed at all in IPv6 profiles. For IPv4, all profiles could be designed in three variants depending on

the ID handling. 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 profiles with more flexible mechanisms requiring less bits for the ID handling as long as sequential jump assignment is used. Such profiles 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, profiles could be designed without any additional information for the ID field included in compressed headers. To use such profiles, 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 profiles is very uncertain. However, designers of IPv4 stacks for cellular terminals SHOULD use the sequential policy.

7.2.2. 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.

7.2.3. 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.

7.2.4. UDP Checksum

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

In this document, there are profiles defined for packet streams both with and without support for the UDP checksum. Profiles with this support will of course require more bits to be sent in compressed headers.

7.2.5. 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.

7.2.6. RTP Marker

The marker bit should be set only in the first packet of a talkspurt, which means that it has a semistatic characteristic with well-known values for both states.

7.2.7. 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.

7.2.8. RTP Sequence Number

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

7.2.9. RTP Timestamp

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.

7.2.10. 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.

7.3. 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.

7.3.1. List of defined profiles

All defined profiles are listed in Table 7.2 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 5.4 and "Header compression profile" in the Terminology chapter.

The first six 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.

CPS This column gives the number of Concurrent Packet Streams that are supported by the header compression profile.

Chk This column indicates whether the profile supports packet streams with the UDP checksum (E)nabled or D(isabled). For IPv6, which prohibits disabling of the checksum, a third kind of profiles should be defined (C)ompressing the checksum.

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 cannot handle streams with IP Identification values assigned using any other policy than sequential.
(S)EQUENTIAL (J)UMP	These profiles can handle all kinds of Identification assignment methods but will be less efficient than RANDOM profiles if the assignment truly is random. "+" suits frequent jumps.
(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.

S/R S gives the minimal header Size for the profile while R represents a Robustness value. R corresponds to the number of packet losses that can be handled without losing context.

The next five columns indicate how each profile is implemented. This includes header formats for STATIC (STA, chapter 7.5.1), DYNAMIC (DYN, chapter 7.5.2) and COMPRESSED (COM, chapter 7.5.3) packets, and also what EXTENSION (EXT, chapter 7.5.5) formats are used with the COMPRESSED packets and the interpretation of the Code-field (CFI, chapter 7.6).

Nr	IPv	CPS	Chk	Id	S/R	STA	DYN	COM	EXT	CFI
1	6	1	E	-	4/26	1	5	2	A	S
2	6	28	E	-	4/5	2	6	4	A	C
3	6	256	E	-	5/26	2	6	6	A	S
4	4	1	D	S	2/26	3	3	1	A	S
5	4	1	E	S	4/26	3	7	2	A	S
6	4	28	D	S	2/5	4	4	3	A	C
7	4	28	E	S	4/5	4	8	4	A	C
8	4	256	D	S	3/26	4	4	5	A	S
9	4	256	E	S	5/26	4	8	6	A	S
10	4	1	D	SJ+	2/5	3	3	7	B	I/S
11	4	1	E	SJ+	4/5	3	7	8	B	I/S
12	4	256	D	SJ+	3/5	4	4	9	B	I/S
13	4	256	E	SJ+	5/5	4	8	10	B	I/S
14	4	1	D	SJ	2/26	3	3	7	B	S/I
15	4	1	E	SJ	4/26	3	7	8	B	S/I
16	4	256	D	SJ	3/26	4	4	9	B	S/I
17	4	256	E	SJ	5/26	4	8	10	B	S/I
18	4	1	D	R	4/26	3	3	11	A	S
19	4	1	E	R	6/26	3	7	12	A	S
20	4	28	D	R	4/5	4	4	13	A	C
21	4	28	E	R	6/5	4	8	14	A	C
22	4	256	D	R	5/26	4	4	15	A	S
23	4	256	E	R	7/26	4	8	16	A	S
24	4	1	D	S	1/16	3	3	1M	A	S

Table 7.2 : List of defined profiles

7.3.2. Additional common profile characteristics

In addition to what was stated in the left part of Table 7.2, the following applies to all the profiles defined in this document:

Packet stream characteristics

These profiles are designed for packet streams carrying IP telephony data.

Link layer requirements

Except for the general link layer requirements (framing, length & profile negotiation) stated in chapter 5.6, 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 on a single "guess" handle loss of up to 26 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.

7.4. Encoding methods used

The analysis in section 7.2 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 7.2, 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 three different methods of handling the sequence number field, LSB encoding, LSP encoding, and reconstruction attempts based on "normal case" knowledge. The first two are also used for

encoding of a variety of other fields, and this chapter therefore describes these methods in a general way. How these two methods are applied for different profiles and how the method with "normal case" reconstruction attempts works is described in chapters 7.6 and 7.7.

7.4.1. 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.

7.4.2. Least Significant Part (LSP) encoding

One restriction with LSB encoding is that an exact number of 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.

7.5. 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 in all packets are reserved. These patterns are:

```
STATIC       00000
DYNAMIC      0001*
FEEDBACK     00001
```

The other 28 (32-4) bit patterns indicate a COMPRESSED packet format and the usage of these patterns are explained in chapter 7.6 and chapter 7.7. These five bits are hereafter referred to as the Code-field in COMPRESSED headers.

Note that for profiles using the MINIMAL_COMPRESSED header sub-type described in chapter 7.6.4, all bit patterns starting with a "1" are ALSO reserved for identification of this header type.

```
MINIMAL_COMPRESSED  1****
```

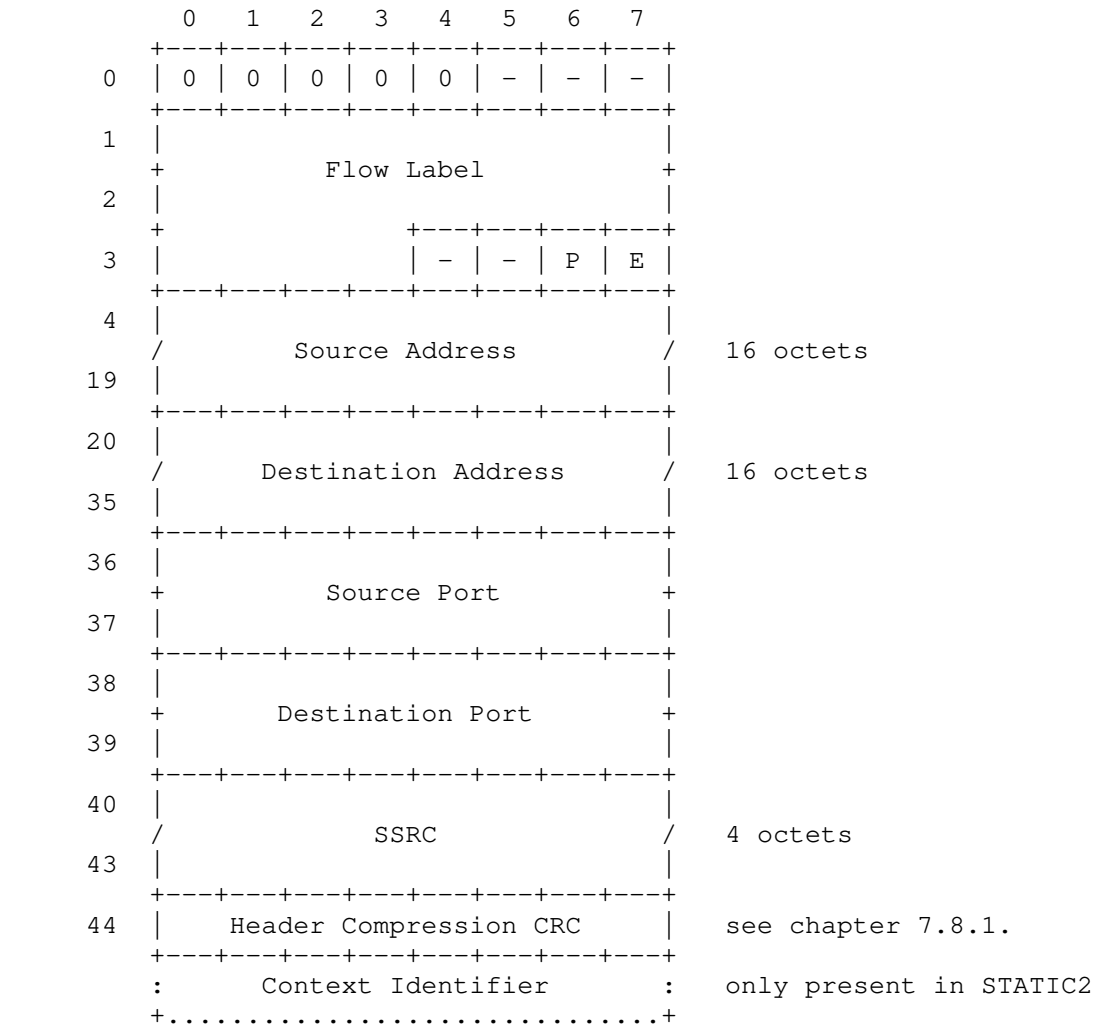
That leaves 12 (16-4) bit patterns in the Code-field for other usage in the "normal" COMPRESSED header for such profiles.

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. Subsections are also dedicated to the MINIMAL and EXTENSION formats of COMPRESSED headers.

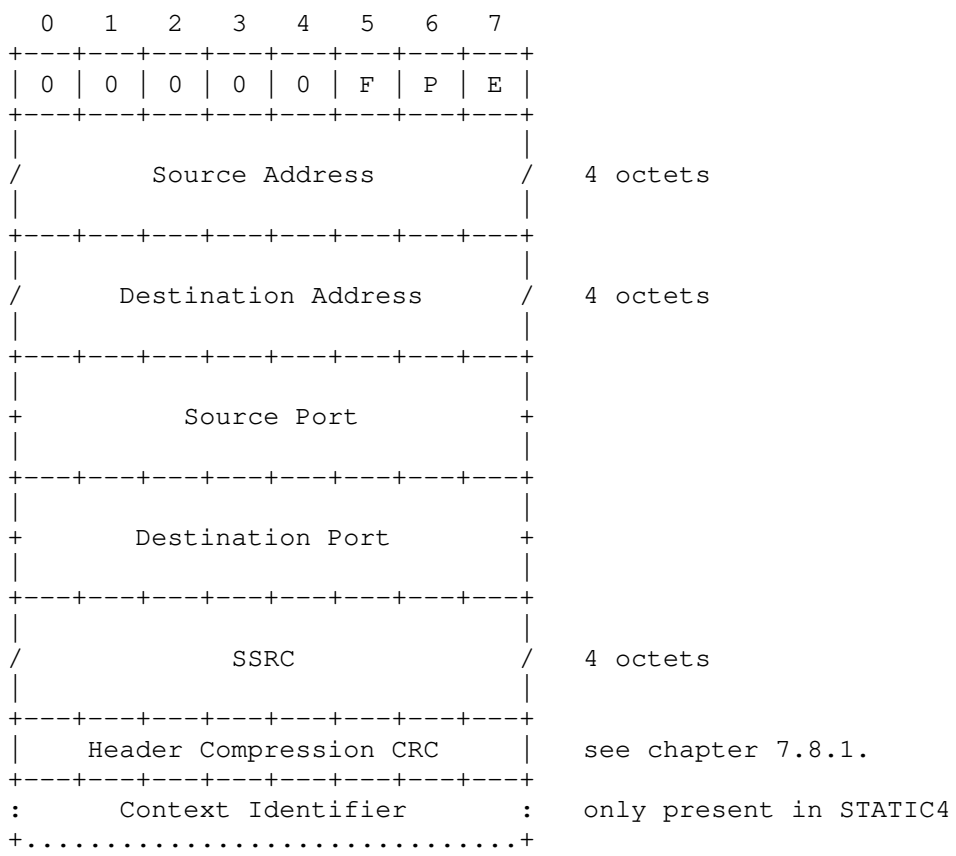
7.5.1. Static information packets, initialization

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 chapter 6). 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 7.6.7 and 7.7). 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:



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



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.

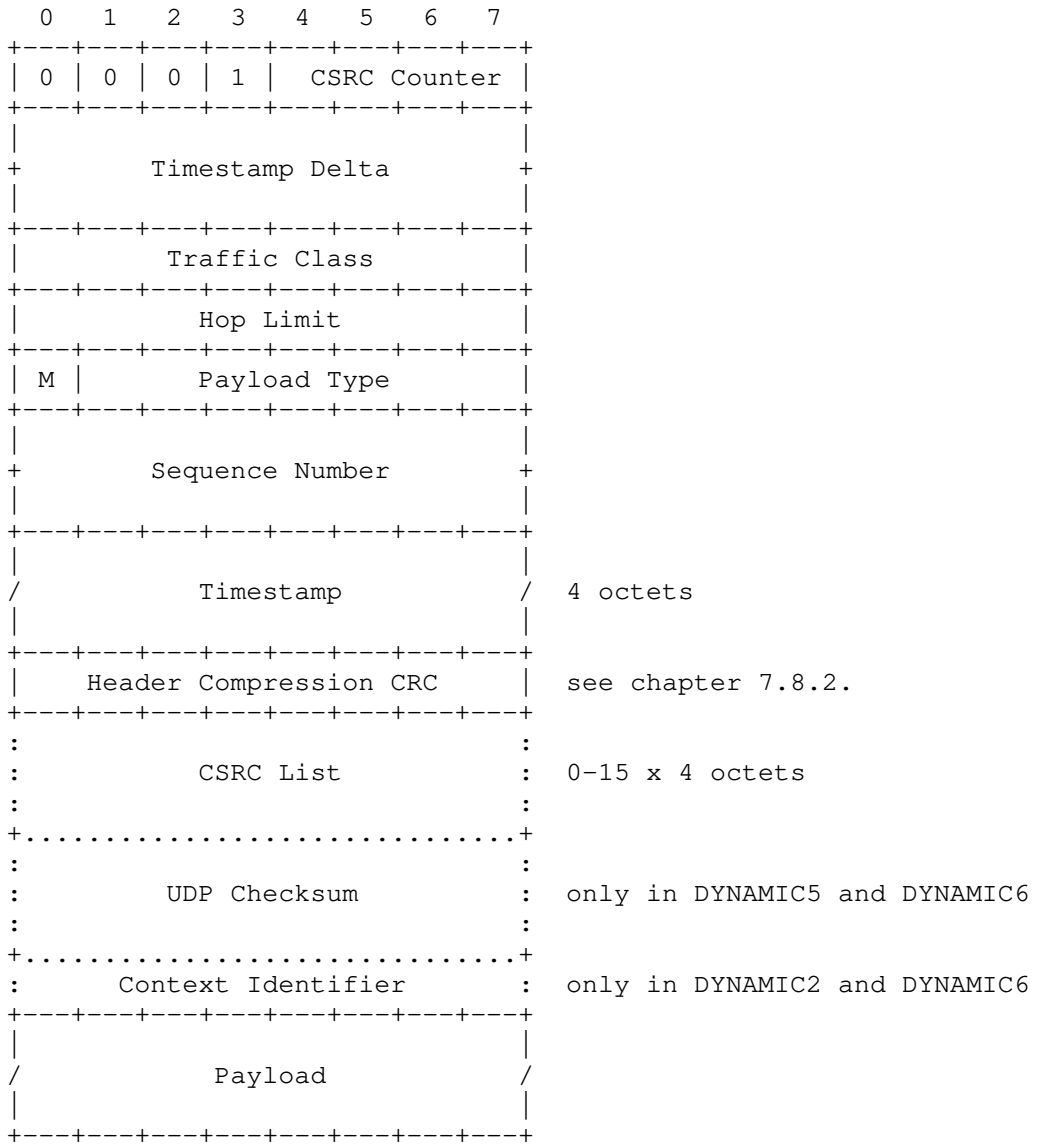
7.5.2. Dynamic information packets

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.

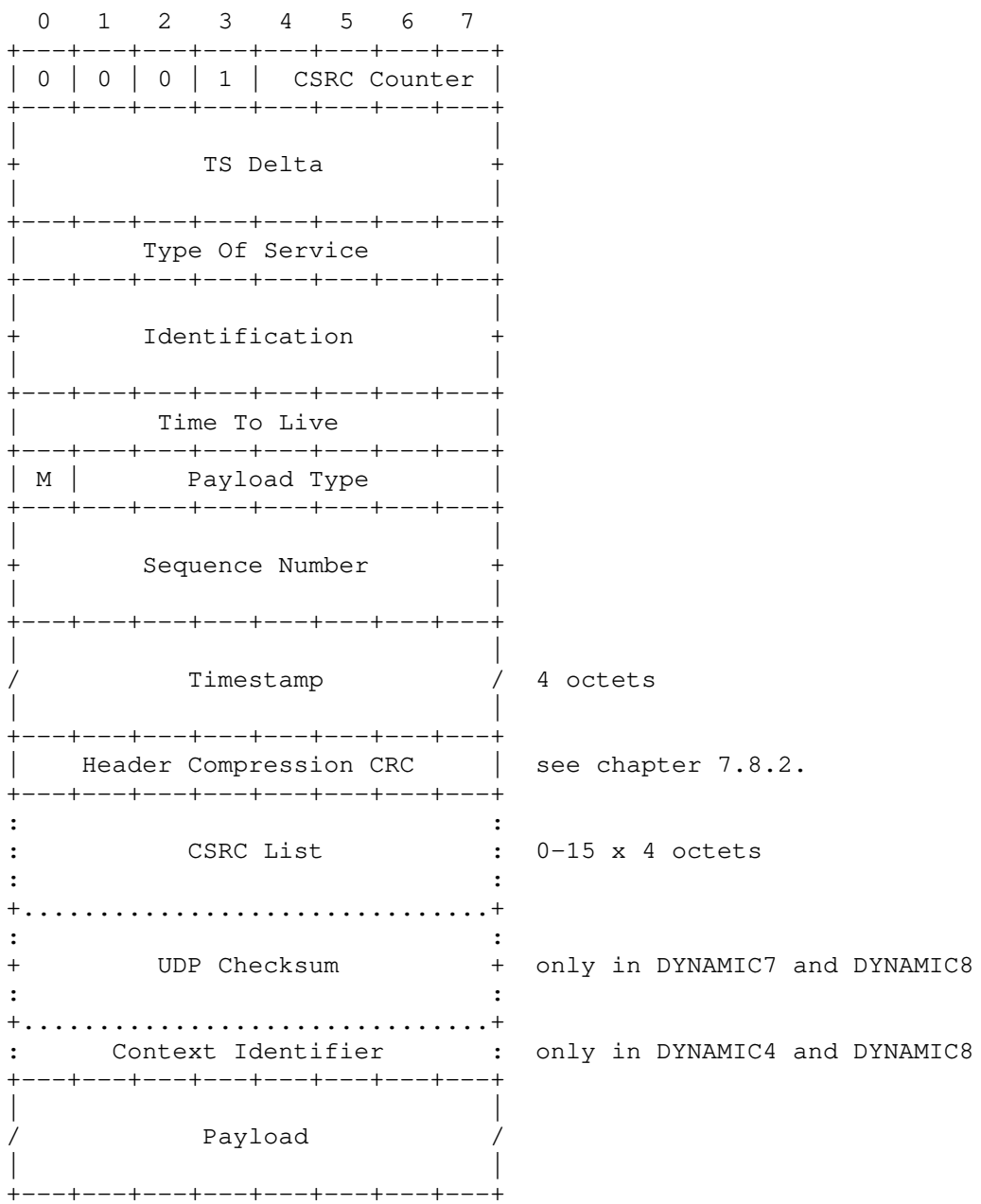
All fields except for the initial four bits, the Timestamp Delta, and the Header Compression CRC are ordinary IP, UDP and RTP fields. The Timestamp Delta is the current delta between RTP timestamps in consecutive RTP packets. Initially this value SHOULD be set to 160.

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, DYNAMIC5, DYNAMIC6:



IPv4 (15-18 octets + CSRC List of 0-60 octets): DYNAMIC3, DYNAMIC4, DYNAMIC7, DYNAMIC8:

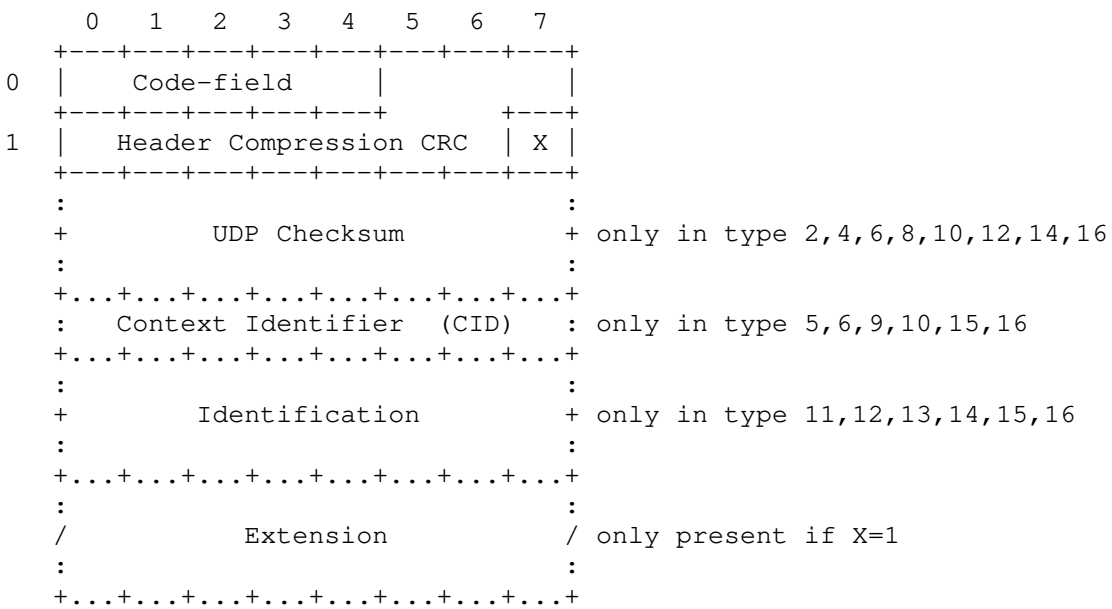


Each time a DYNAMIC packet is sent, several subsequent packets SHOULD also be DYNAMIC packets to ensure a successful update even when packets are lost. Context updates both with DYNAMIC and COMPRESSED packets could also be acknowledged with CONTEXT_UPDATED_FEEDBACK.

7.5.3. 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, with only the Header Compression CRC of 10 bits, the Code-field and one bit indicating if header extensions are present. 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..COMPRESSED16:



The Header Compression CRC is computed over the original packet header as described in chapter 7.8.3. In that chapter, the CRC polynomials to use are also defined.

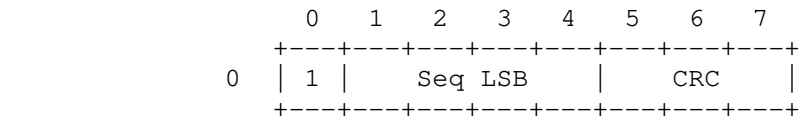
The interpretations of the Code-field for different profiles are given in section 7.6 and section 7.7.

7.5.4. Minimal compressed headers

Profiles using COMPRESSED packet types marked with an "M" also support a special MINIMAL_COMPRESSED header type in addition to the ordinary one. This header type is indicated for such profiles by setting the first bit to 1, while 0 means that the normal compressed

header type is used. MINIMAL_COMPRESSED headers SHOULD only be used when header field changes are regular before the compression point, otherwise the normal compressed format SHOULD be used.

MINIMAL_COMPRESSED packet type (used with COMPRESSED*M):



Sequence LSB is included as described in chapter 7.7.2. The CRC is a reduced form of the Header Compression CRC used in ordinary compressed headers. The CRC polynomial to be used is defined in appendix B. The small CRC does not provide a very high reliability. Therefore, when using the MINIMAL_COMPRESSED packet type, it is recommended to implement the CRC pre-verification mechanism described in chapter 8.3 to reduce this weakness to an insignificant level.

Note that because of the need to identify the minimal compressed header type, the Code-field interpretation is different than for profiles without this header type, as described in 7.7.1.

7.5.5. Extensions to compressed headers

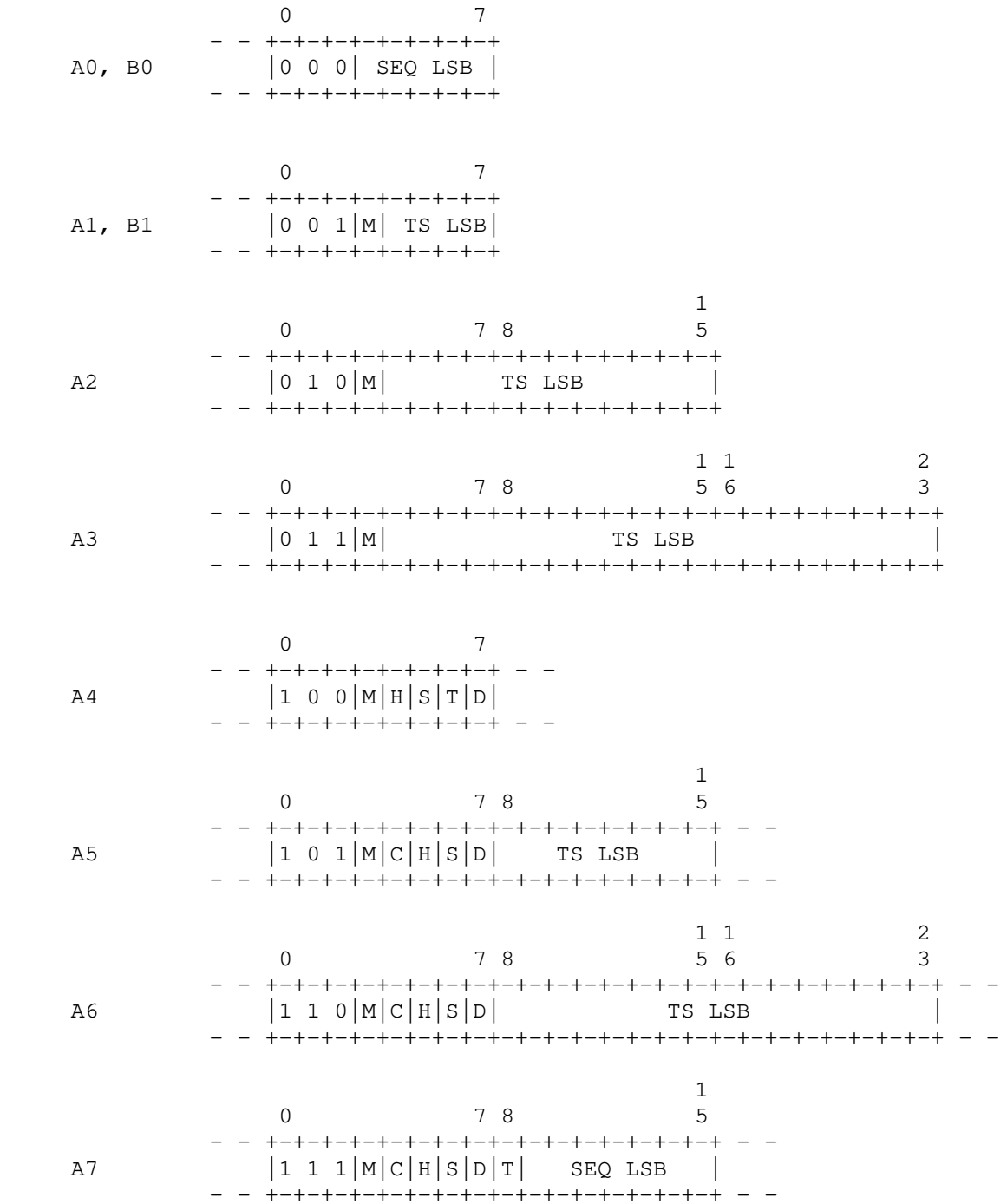
Less regular changes in the header fields or updates of decompressor contexts require an extension in addition to the base header. When there is an extension present in the COMPRESSED packet, this is indicated by the extension bit (X) being set. Extensions are of variable size depending on the information 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 TS LSB field, a SEQ LSB field, an ID LSB field and a bit mask for additional fields. The M bit is the RTP marker bit and the TS LSB is the least significant bits of the timestamp value (the most significant bits are then expected to be unchanged since previous packets). The SEQ LSB (described in chapter 7.7.2) is the least significant bits for the RTP sequence number, and 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.

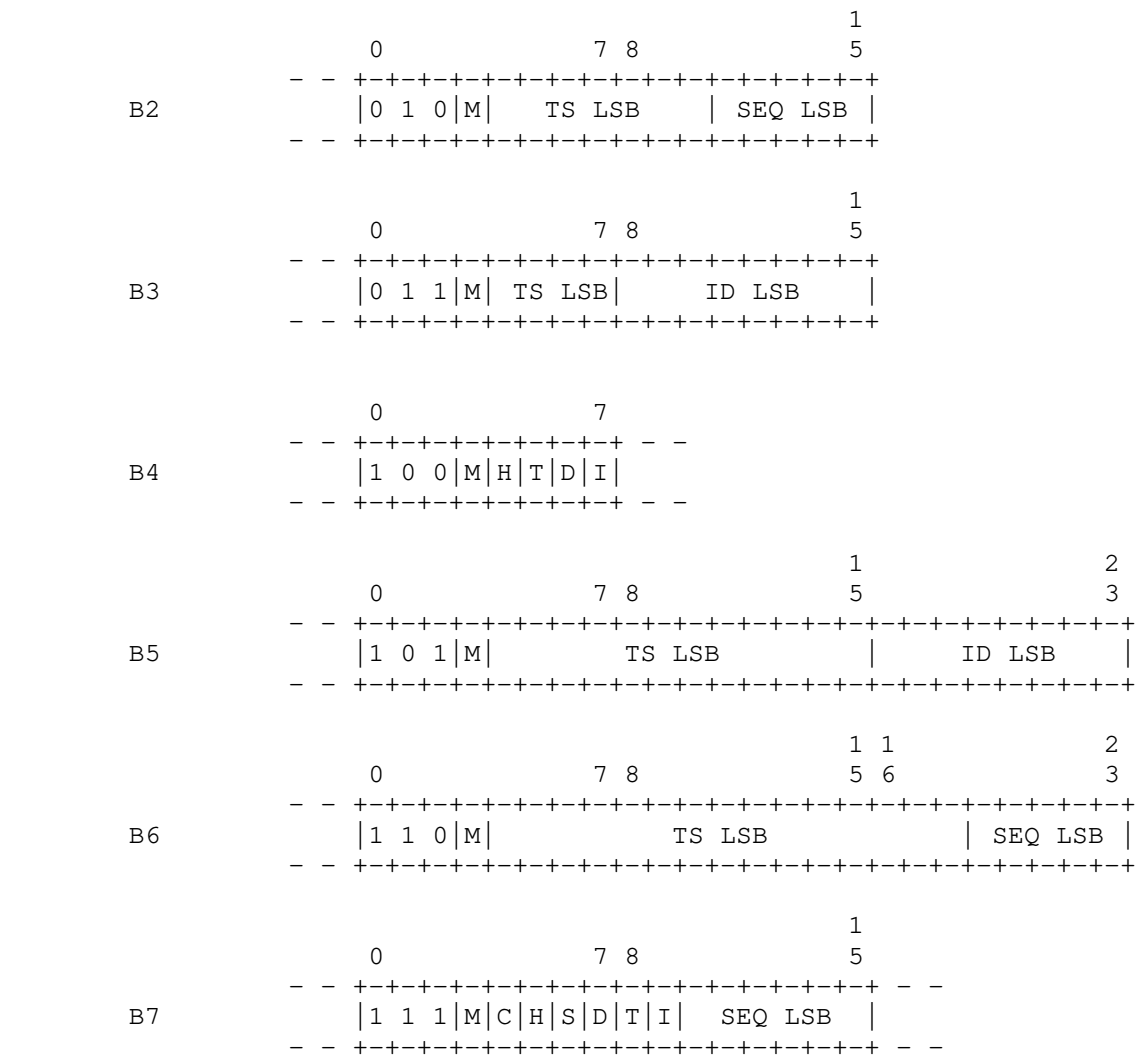
The guiding principle for choosing the extension type is to find the smallest header type that can communicate the information needed.

For the profiles defined in this document, two different extension sets are used, called A and B. Set A is the simpler one while set B handles much more functionality and is therefore more complex. 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:





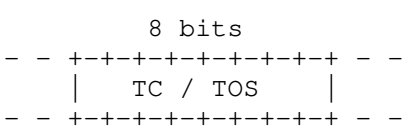
A bit mask indicating additional fields could include bits with the following meaning:

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.

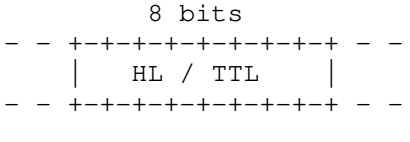
C - Traffic Class / Type Of Service

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



H - Hop Limit / Time To Live

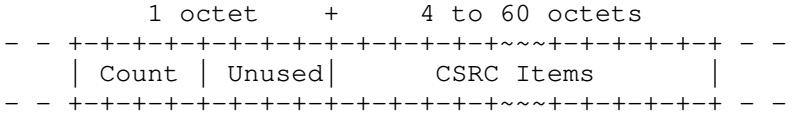
The field contains the value of the original IP header field.



S - Contributing Sources

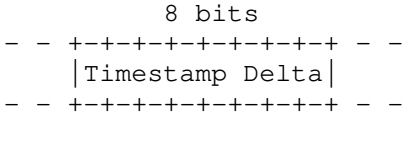
The CSRC field is built up of:

- 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)



D - Timestamp Delta

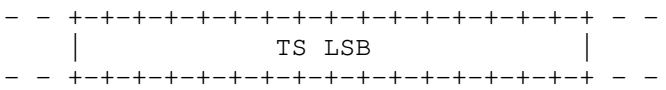
The Timestamp Delta field is a one-octet field. We want to communicate Timestamp Delta values corresponding to 80 ms. Therefore, the Timestamp Delta value communicated is not the actual number of samples, but the number of samples divided by 8. Thus, only Timestamp Delta values evenly divisible by 8 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.



Note that when the Timestamp Delta is changed, the Timestamp LSB field MUST also be included.

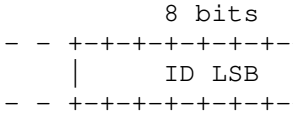
T - Timestamp LSB

The field contains the 16 least significant bits of the RTP timestamp.

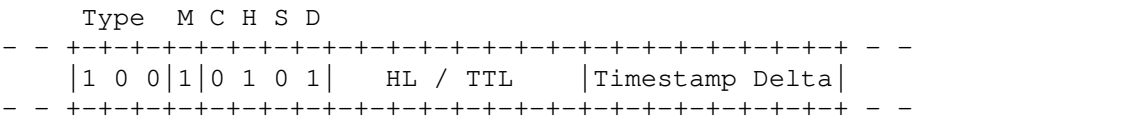


I - Identification

The field contains the IP ID LSB.



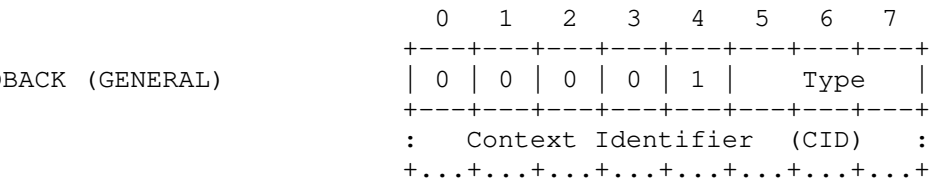
An example where the HL/TTL and the Timestamp Delta fields are present in a type A4 extension is shown below. When the Timestamp Delta field is present the RTP Marker will probably also be set, which is the case in this example.



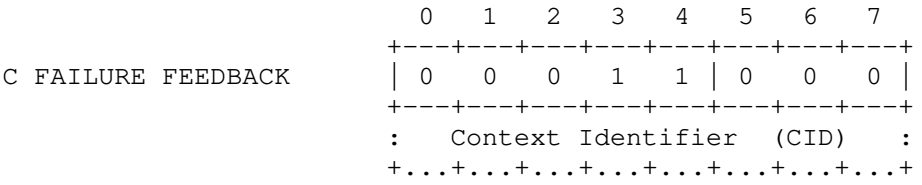
When information of any kind is sent in an extension, the corresponding information SHOULD also be sent in some subsequent packets (either as Extensions or in DYNAMIC packets).

7.5.6. Feedback packets

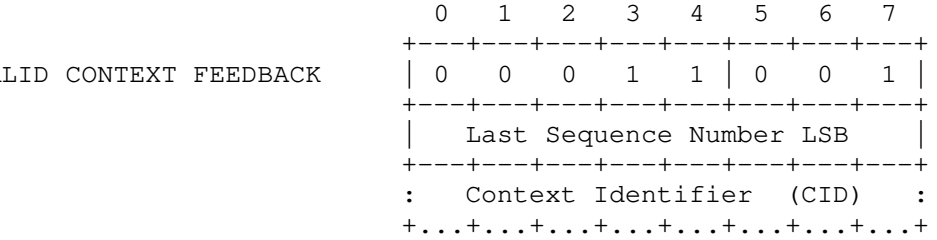
Feedback packets are used by the decompressor to provide various types of feedback to the compressor. That could include active feedback to assure error free performance or passive feedback (in case of invalidated context) to request a context update from the compressor. The feedback mechanisms defined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



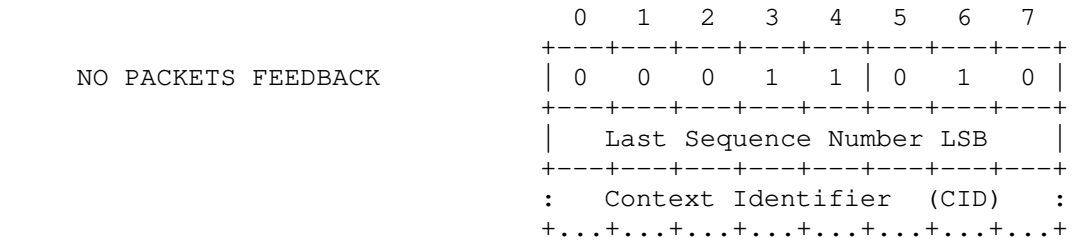
Note that The CID field is present only for profiles using STATIC packet format 2 or 4, which are profiles supporting multiple packet streams. The Type field tells what kind of feedback the packet corresponds to and the feedback types defined are the following:



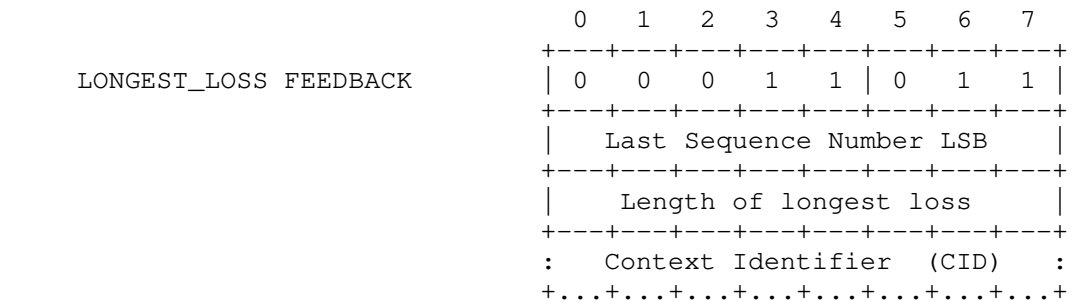
The STATIC_FAILURE_FEEDBACK packet tells the compressor that the static part of the decompressor context is invalid, and that an update of that part is required. Reasons for sending such feedback could be that no STATIC packet has been received at all, or that decompression has failed even when DYNAMIC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



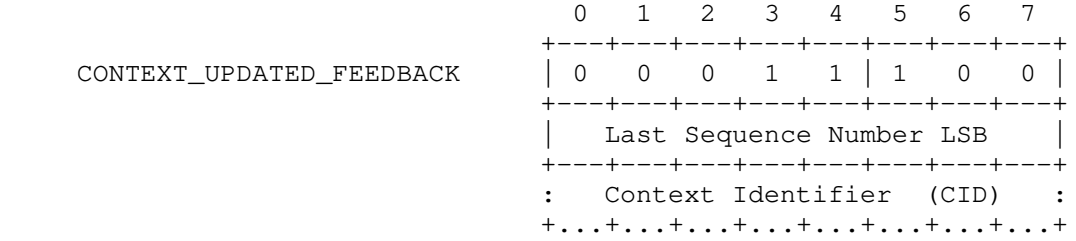
The INVALID_CONTEXT_FEEDBACK packet SHOULD be sent to signal an invalid decompressor context, indicated by failing decompression of COMPRESSED packets.



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.



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.



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.

7.6. Interpretations of the Code-field

The usage of the Code-field in COMPRESSED headers (the 28 or 12 code-points left after packet type identification) differs from profile to profile. The field is used in four different ways for the profiles defined in this document, as shown in table 7.2.:

S - Sequence encoding

The remaining 28 or 12 code-points are used for sequence number LSP encoding as described in chapter 7.7.1.

C - Context Identification (CID)

The code-points are used to separate up to 28 different packet streams. When used in this way, the sequence encoding MUST be done in extension headers. The encoding is performed using the same principles as for S above, but with LSB instead of LSP and using the SEQ LSB field of an extension. However, as long as the sequence value is increasing by one and also has been in at least the three preceding packets, the sequence information MAY be completely omitted. If the decompressor receives a packet without sequence information it uses reconstruction attempts to find the correct value, as described in chapter 7.7.3.

I/S - IP Identification LSP OR Sequence encoding

For profiles using this Code-field interpretation, the field could be used both for sequence and IP Identification encoding. There are two rules for where to sent what:

Normally, the IP Identification LSP is encoded in the Code-field as described in chapter 7.7.1. The sequence number is then either sent as the LSB in an extension header or not sent at all as described in chapter 7.7.3.

If the IP identification offset has changed too much to be encoded in the Code-field, it MUST be sent in an extension instead as described in chapter 7.7.2, and for these cases the sequence number MUST be encoded in the Code-field as described in 7.7.1.

S/I - Sequence encoding OR IP Identification LSP

This is almost the same interpretation as in I/S above but with the difference that as long as the IP Identification follows the sequence number, sequence LSP is encoded in the Code-field. IP Identification LSP is sent in the Code-field only if the sequence number LSB is sent in an extension (provided that the number of code-points are sufficient).

7.7. 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 numbers with three different methods: LSP encoding, LSB encoding and, in some common cases, header reconstruction attempts without requiring any information in the compressed header. LSP and LSB encoding are also used for the IP Identification field in some of the profiles defined here. This chapter describes how the encoding methods in chapter 7.4 are applied to the various field values.

7.7.1. LSP encoding of field values

LSP, as described in chapter 7.4.2, is used in the Code-field of the "normal" COMPRESSED header (chapter 7.5.3). For profiles not using the MINIMAL_COMPRESSED header format, there are 28 code-points in the Code-field left for sequence encoding. An LSP of size 28 is therefore used with the following encoding (+4 because 0-3 are reserved):

```
CODE(n) = LSP:28(n) + 4
```

For profiles using the MINIMAL_COMPRESSED header format, 12 code-points in the "normal" COMPRESSED header are left for other usage, meaning that an LSP of size 12 must be used instead. However, the encoding is the same:

```
CODE(n) = LSP:12(n) + 4
```

The reordering parameter for LSP MUST be set to 1 meaning that first order reordering can be handled by encoding in the Code-field.

7.7.2. LSB encoding of field values

In MINIMAL_COMPRESSED headers (chapter 7.5.4), LSB encoding with 4 bits is used for sequence numbers. LSB encoding with various sizes is also used when transmitting sequence number, timestamp and IP Identification information in extended compressed headers.

7.7.3. Sequence encoding with no information

Some profiles make use of headers without any sequence number information at all provided. Such headers MUST NOT be used by the compressor if the sequence number has changed irregularly (by other values than +1) for the current or any of the 3 previous packets transmitted. In such cases, extended compressed headers MUST be used. If the decompressor receives a packet without explicit sequence information, it should instead use reconstruction attempts to find the correct value. The attempts are based on the knowledge that the received and previous sent (maybe lost) packets all had a sequence number increase of 1. The attempts SHOULD be:

1 - No loss: S(n) = S(n-1) + 1

2 - One lost: S(n) = S(n-1) + 2

3 - Two lost: S(n) = S(n-1) + 3

4 - Three lost: S(n) = S(n-1) + 4

If possible, more attempts could be performed.

7.8. 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.

7.8.1. 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:

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

7.8.2. 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:

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

7.8.3. COMPRESSED packet CRCs

The header compression CRC in the COMPRESSED header is calculated over the same headers as the CRC in the DYNAMIC packet. The only difference is that the polynomial to be used is:

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

In the MINIMAL_COMPRESSED header, the coverage is the same but with a different polynomial:

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

8. 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.

8.1. Feedback and context update procedures

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 when the implementation experience grows.

8.2. 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 transport systems that cannot possibly send information back to the compressor. The most important case may be when the packet are broadcast in some way. It can, for

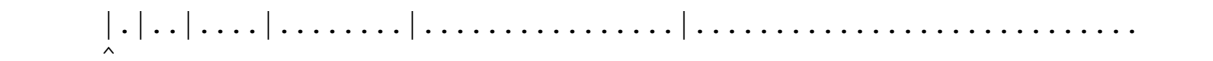
example, be the communication from a satellite. Over a simplex link the decompressor does not have the possibility of sending information back to the compressor. The compressor does not know when the decompressor needs a STATIC or DYNAMIC packet. If STATIC and DYNAMIC packets are sent at regular intervals 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 guesses 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.

8.2.1. 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:



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

8.2.2. 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.

8.2.3. 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.

8.2.4. Cost and robustness of refreshes

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

$$\frac{(S+U-C)}{f} + C$$

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:

$$\frac{(S+U-C)}{f}$$

If we assume that we use ROCCO profile number 4 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 (appendix B.6). 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 4 it is possible to lose up to 26 consecutive packets without losing the context in the decompressor. The probability that 27 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.

8.2.5. 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.

8.3. 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.

8.4. Using "guesses" with LSB and LSP encoding

ROCCO profiles using LSP encoding can handle 26 or 10 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 7.4.2, 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.

9. Further work

The ROCCO scheme, including the compression profiles for IP telephony 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.

9.1. Timer-based timestamp reconstruction

The RTP timestamp field is one of the header fields that may change dynamically on a per packet basis. In chapter 7.7 it is stated that the timestamp value can be inferred from the encoded RTP sequence number value for audio services 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 7.5.5 extension headers are used to transfer this leap in the timestamp value. That increases, however, the average header size. This subchapter presents a possible non-mandatory mechanism for solving this problem, without adding any additional header bits, through the 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. 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)

In a numeric example with dT = 103 ms, this translates into:
timestamp change = round(103/20) * 160 = 5*160 = 800
Hence, the local clock at the decompressor indicates that the timestamp has changed by 5 packet time intervals and the timestamp should thus be incremented by 5*160 = 800.

This timestamp change value is finally added to the previous timestamp value (as received at time T0) to give a reconstructed value of the timestamp.

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 correctness of a timestamp estimation based on timers MUST always be verified to ensure the correctness of the decompressed full header. This verification is preferably done using the header compression CRC or possibly some small number of least significant timestamp bits included in the compressed header.

This mechanism makes it possible for the decompressor to reconstruct the timestamp value at the beginning of speech periods (when the timestamp has taken a leap compared to the sequence number) without imposing the need for extra timestamp bits in the compressed header at this event. Hence, it removes the need for sending extensions to compressed headers when the timestamp value cannot be inferred from the sequence number and therefore decreases the average header size.

The usefulness of timer based timestamp reconstruction may be smaller for video services since the timestamp for video services will have a less regular timestamp increment compared to the sequence number increment. If, for example, MPEG4 is used, the timestamp may in fact have a negative difference compared to the previous packet. MPEG4 can use I, P and B-frames. B-frames are encoded using two consecutive P-frames and the B-frame contains the video contents between these two P-frames. The B-frame is however sent (with RTP) after the two P-frames, and the timestamp difference may thus be negative compared to the sequence number change. Hence, a timer-based timestamp reconstruction may thus have a higher failure rate for video services.

The usage of timer-based timestamp reconstruction MUST NOT be mandatory in a scheme, since a decompressor should not be dependent on the availability of a wall clock or timer.

9.2. 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.

9.3. Replacement of the UDP checksum

When the UDP checksum is enabled (which is always the case with IPv6), it must be included in all compressed headers, meaning that the minimal header size is increased by 2 octets. This is undesirable from a compression point of view but for most header compression schemes there has been no other solution.

However, there is one other possible solution which is applicable especially to ROCCO. The idea is to replace the UDP checksum by a stronger checksum over the link where header compression is carried out. A CRC would provide a stronger error detection than the UDP checksum, even with fewer bits, and if the CRC coverage is chosen in a suitable way it could at the same time serve as the header compression checksum. By combining the UDP checksum and the header compression CRC into one stronger CRC, header compression profiles could be designed with a smallest header size of 3 or maybe only 2 octets, even with support for the UDP checksum.

This idea would be even more applicable in a scenario where UDP Lite is used with a checksum coverage limited to the headers only, because the checksum coverage would then be exactly the same as for the header compression CRC in ROCCO.

9.4. 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.

9.5. General, media independent profiles

This document defines header compression profiles optimized for IP telephony packet streams. Independently of packet stream characteristics, these profiles will successfully compress and decompress the headers of all IP/UDP/RTP packets. However, the compression will 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.

10. 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. Currently, profile #7 and #23 of ROCCO version 03 (renumbered #4 and #24 in ROCCO version 04) are implemented. 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 slightly more computational power than CRTP. On the other hand, with ROCCO the audio quality is significantly better. Figure 10.1 shows a block diagram of the testbed environment.

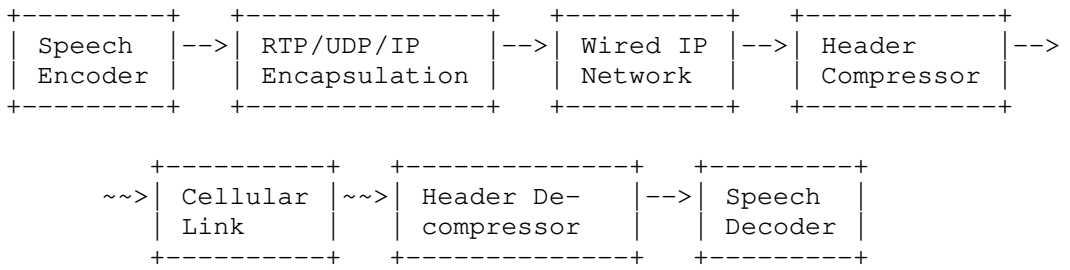


Figure 10.1 : Block diagram of testbed environment

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/

11. 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 IP telephony 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 26). 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 forwarding of erroneous headers.

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 Appendix B, 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.

12. 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.

13. Acknowledgements

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

Thanks to Anton Martensson for many valuable draft contributions.

Andreas Jonsson (Lulea University) made a great job supporting this work in his study of header field change patterns. Our collaboration on consistent field classification methods was also very fruitful and resulted in great improvements to those parts of this document.

14. Intellectual property considerations

This proposal in is conformity with RFC 2002.

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.

15. 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, July 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), October 1999. <draft-westberg-realtime-cellular-01.txt>

[ROVID] Anton Martensson, Torbjorn Einarsson, Lars-Erik Jonsson, "ROCCO Conversational Video Profiles", Internet Draft (work in progress), March 2000. <draft-martensson-rocco-video-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. http://www.etsi.org

16. 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 (According to chapter 6)

In this appendix, all IP, UDP and RTP header fields are classified as STATIC, STATIC-DEF, STATIC-KNOWN, INFERRED or CHANGING. For all fields except for those classified as CHANGING, the classifications are also motivated. CHANGING fields should be further classified based on their expected change behavior for various kinds of packet streams.

A.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.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.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.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.5. Summary

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)

Appendix B. Simulated performance results

To evaluate the performance of ROCCO and the IP telephony profile, two header compression schemes have been simulated; CRTP [CRTP] and ROCCO, both the 2 octet solution (profile number 4) and the 1 octet solution (profile number 24). Sections B.1 to B.5 provide the parameters used in the simulations. Sections B.6 and B.7 show the results.

B.1. Simulated scenario

A source generates RTP packets, which are sent over a wired network to an end-system where the last link is a cellular link. The RTP stream is compressed over the last cellular link using one of the two header compression schemes. The simulated scenario is shown in Figure B.1.

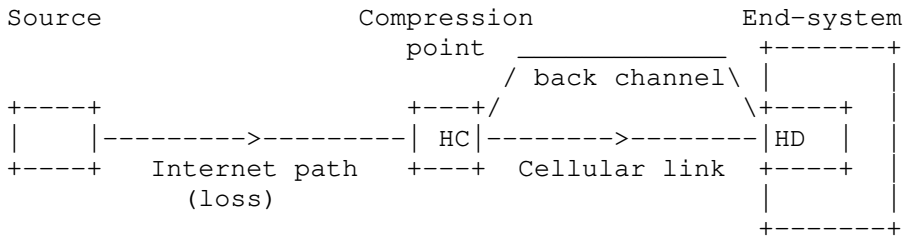


Figure B.1 : Simulated scenario.

B.2. Input data

The speech source generates packets with a fixed size, 244 bits, every 20 ms (12.2 kbps), corresponding to the GSM enhanced full-rate speech codec. On top of these bits, there is a 12 bit application CRC, making up a total of 256 bits (32 octets).

The length of the talk spurts and the silent intervals between them are both exponentially distributed with an expected length of 1 second. Silence suppression is used, meaning that no data is transmitted during silent periods.

B.3. Influence of pre-HC links

The packet stream suffers from a 0.5% uniformly distributed packet loss, i.e. in a prior IP network. There is no reordering of packets. The purpose of using high error probabilities is to test the capabilities of the schemes also under rough conditions.

B.4. Used link layers

CRTP needs to have the packet type identification provided by the link layer, whereas ROCCO has the packet type identification integrated. Hence one octet extra of link layer overhead is added for the CRTP case. This octet is not included in the presented result. A 16 bit CRC covers only the header and not the payload. This fits in well with speech coders, which can conceal some damage. This will also increase the headers seen by the decompressor and hence improve header compression performance.

A packet is considered as lost if it is not passed up to the application (speech codec). There are three possible reasons for packet loss in these simulations:

1. A bit error has occurred in the compressed header.
2. A bit error has occurred in the link layer packet type identification (for the CRTP case only) or in the link layer checksum.
3. The header compression scheme has a faulty context and cannot decompress any received compressed header (context damage). Note that this can happen even if the compressed header is error free.

B.5. The cellular link

The cellular link is a WCDMA channel simulated with the fading model in [WCDMA]. The reported bit error rates, BER, are the BERs seen by the link layer and thus the BERs after channel coding. The back channel used in our simulations never damages the FEEDBACK messages. The RTT is set to 120 ms.

B.6. Compression performance

Figure B.2 shows the average header size plotted against BERs for the two header compression schemes. For BERs around 1e-4 CRTP and ROCCO profile 4 gives an average header size of just above 2 octets (2.15 for both). ROCCO profile 24 has an average header size just above 1 octet, 1.20, at the same BER. The average header size for CRTP starts to increase when BER becomes slightly larger than 1e-4; at BER 1e-3 it is 2.35. At higher BERs than 1e-3 the average header size for CRTP increases faster, and at 1e-2 it is almost 4 octets. For the two ROCCO profiles (4,24) the average header size remains constant at 2.15 octets and 1.20 octets respectively.

The difference between CRTP and ROCCO is mainly that the latter tolerates losing several consecutive packets before it needs a context update packet, while CRTP needs a context update for each loss. ROCCO therefore requires less updates than CRTP introducing less header overhead and losing a significantly lower number of packets.

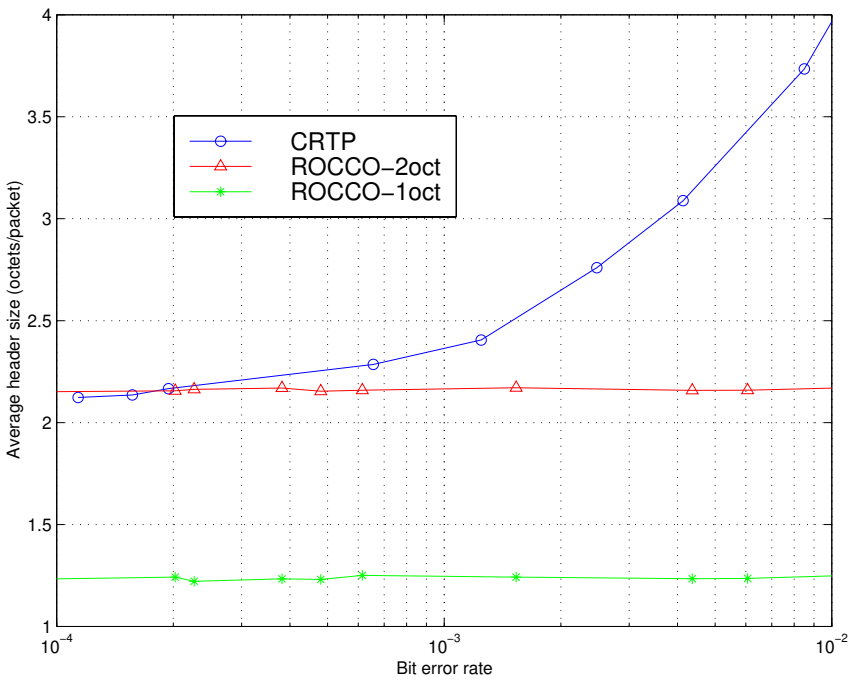


Figure B.2 : Average header sizes

Figure B.3 shows the variation in header sizes for the schemes. The variations are due to silences, and thus the RTP timestamp changes with more than 1 timestamp delta. Most packets are however the smallest available, around 85-95% for both schemes. As ROCCO can handle several consecutive packet losses it never has to make any context requests, but CRTP does have to make them and hence it more often sends larger packets.

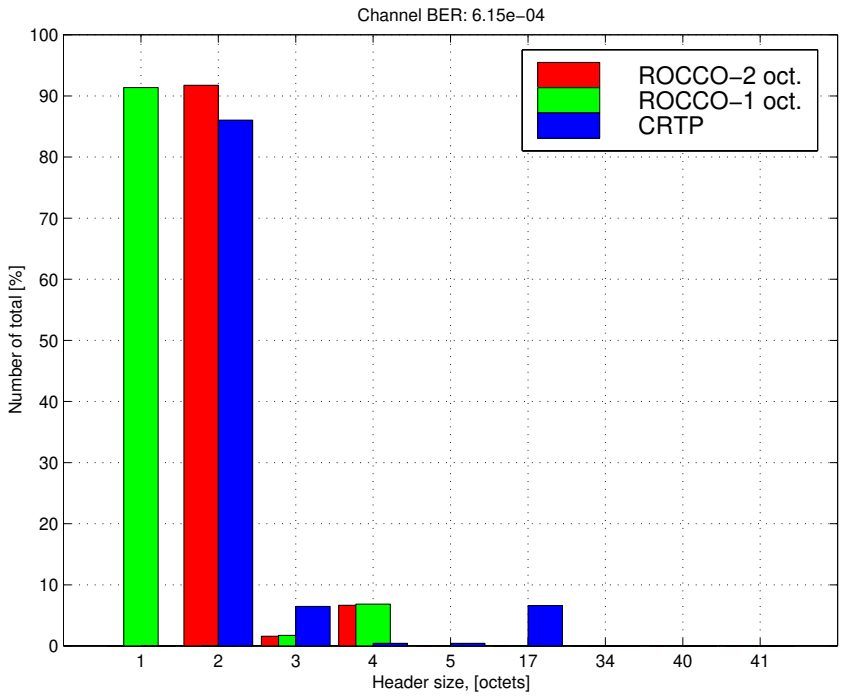


Figure B.3 : Header size variations

B.7. Robustness results

A packet is considered as lost if it is not passed up to the application (speech codec), meaning that a packet with errors in the payload is not regarded as lost as long as it is deemed ok by the header compression scheme.

In figure B.4 the FER is shown for the two header compression schemes. At BER 2e-4 the FER for CRTP is 1.10%; for both ROCCO profiles the FER is 0.12%. When the FER increases to 1e-3, CRTP gives 4.06% FER, ROCCO profile 4 gives 0.81% and ROCCO profile 24 gives 0.69%. The difference in robustness is clearly visible for the two header compression schemes. For CRTP a packet loss between compressor and decompressor triggers a burst of additional losses due to its round-trip based error recovery. In figure B.5 this is clearly visible.

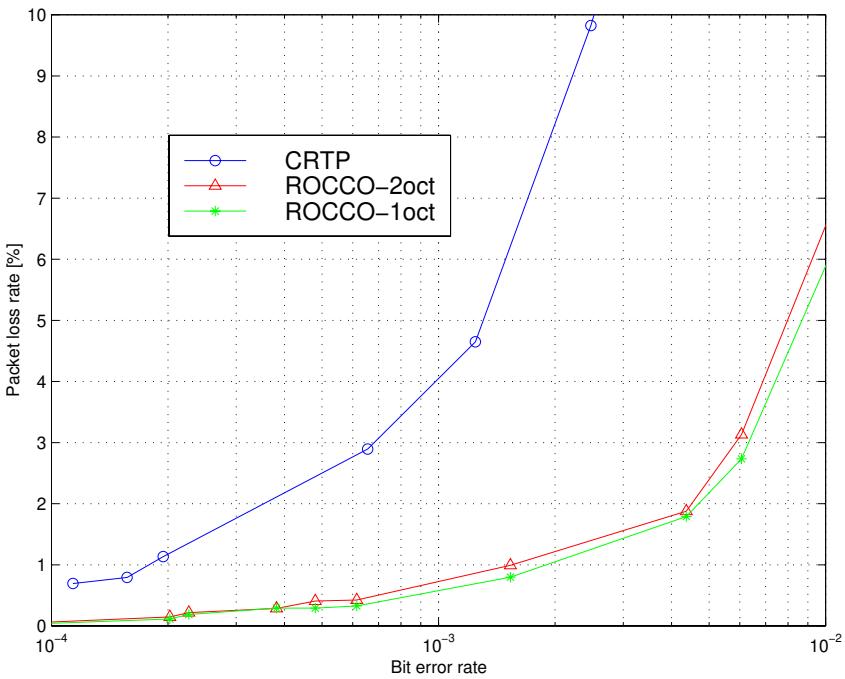


Figure B.4 : Packet loss rate versus BER

Figure B.5 shows the loss pattern for CRTP and ROCCO at BER 4e-3. It is evident from this figure that the majority of the losses are such that 7 consecutive packets are lost. This comes from CRTPs round-trip dependent context repair mechanism. For ROCCO all loss events include 1 or 2 consecutive lost packets, which means that it does not suffer from context damage.

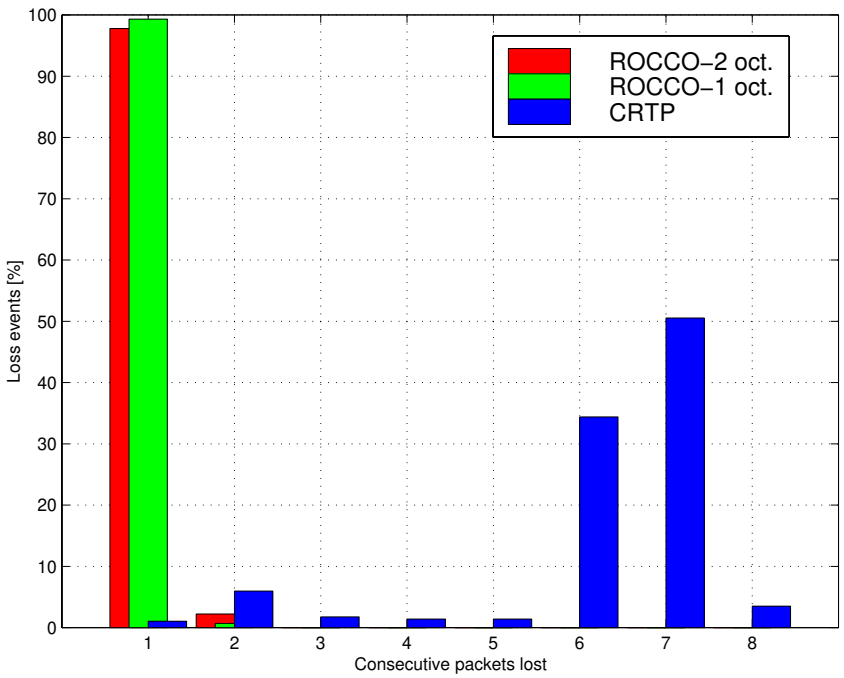


Figure B.5 : Packet loss patterns for CRTP and ROCCO

B.8. CRC strength considerations

The 10 bits of CRC are used to verify guesses when reconstructing the header. The only header fields with bits changing between guesses are the IP identification, RTP sequence number and RTP timestamp. More than 300,000 different combinations of these fields, according to what ROCCO should manage, have gone through a CRC calculation without letting any erroneous packets through. This therefore indicates that 10 bits of CRC is enough to verify the correctness of the guessed header.

This Internet-Draft expires September 10, 2000.