Network Working Group
INTERNET-DRAFT
Expires: April 22, 2000

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

October 22, 1999

RObust Checksum-based header COmpression (ROCCO)
<draft-jonsson-robust-hc-02.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 large.

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....4
2. Terminology....6
3. Existing header compression schemes....8
4. Desired improvements....10
5. Proposed solution....10
5.1. Header compression framework....11
5.2. General ROCCO principles....11
5.3. Data structures....12
5.4. Header compression profiles....12
5.5. Profile negotiation....13
5.6. Link-layer requirements....14
6. Classification of header fields....14
7. Header compression profiles for IP-telephony packet streams..15
7.1. Usage scenarios, environment and requirements....15
7.2. Analysis of change patterns of header fields....16
7.2.1. IPv4 Identification....18
7.2.2. IP Traffic-Class / Type-Of-Service....19
7.2.3. IP Hop-Limit / Time-To-Live....19
7.2.4. UDP Checksum....19
7.2.5. RTP CSRC Counter....19
7.2.6. RTP Marker....20
7.2.7. RTP Payload Type....20
7.2.8. RTP Sequence Number....20
7.2.9. RTP Timestamp....20
7.2.10. RTP Contributing Sources (CSRC)....20
7.3. Profile definitions....20
7.3.1. List of defined profiles....20
7.3.2. Additional, common profile characteristics...23
7.4. Packet types and code field usage....24
7.5. Sequence number encoding....24
7.5.1. Sequence number changes to handle....24
7.5.2. Compression of sequence number values....25
7.5.3. Decompression of sequence number values....26
7.6. Header formats....26
7.6.1. Static information packet, initialization....27
7.6.2. Context update packet....28
7.6.3. Compressed packets....31
7.6.4. Extensions to compressed headers....31
7.6.5. Interpretations of the Code-field....36
7.6.6. Context request packets....37
7.7. Context update procedures....37
8. Simulated performance results....38
8.1. Simulated scenario....38
8.2. Input data....38
8.3. Influence of pre-HC links....38
8.4. Used link layers....39
8.4.1. PPP in HDLC-like framing....39
8.4.2. Link-layer with partial checksum (LLPC)....39
8.5. The cellular link....40

8.6. Compression performance.....40
8.7. Robustness results.....42
8.8. CRC strength considerations.....45
9. Implementation status.....45
10. Security considerations.....46
11. Conclusions.....46
12. Acknowledgements.....47
13. Intellectual property considerations.....47
14. References.....48
15. Authors' addresses.....49

Appendix A. Detailed classification of header fields.....50
A.1. IPv6 header fields.....50
A.2. IPv4 header fields.....51
A.3. UDP header fields.....53
A.4. RTP header fields.....54
A.5. Summary.....55

Appendix B. Mapping of ID and AD values.....56

Document history

00	1999-06-22	First release.
01	1999-09-01	Only small corrections and modifications. The "Sequence Number" field of the "Context Update Header" was erroneously labeled "Destination Port" in the 00 draft.
02	1999-10-22	The concept has been generalized and a number of profiles have been defined with various functionality. New chapters describing profile negotiation, context update procedures, implementation status and security considerations have been added.

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 a revolutionary possibility to always be reachable with reasonable service quality no matter where they are. However, up to 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 (like Internet telephony) has generally been low.

Today, IP-telephony is gaining momentum due to improved technical solutions. It seems reasonable to believe that IP in the years to come will be 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.

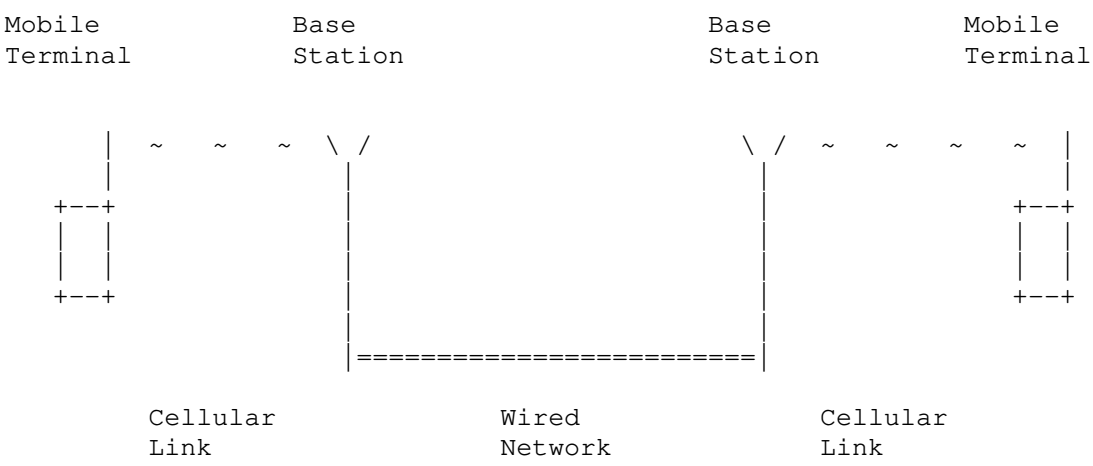


Figure 1.1 : Scenario for IP telephony over cellular links

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 be 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, 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 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.

Ideal header compression scheme

An ideal header compression scheme introduced and defined for comparison purposes. The ideal scheme performs like CRTP would do if used over error-free links to compress input data without irregular changes in its header fields. The compressed header of the ideal header compression scheme is always two bytes, the scheme never loses packets due to context damage, and it does not need to initialize the decompressor context.

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 the original header.

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 is taken 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 are at an acceptable level. A consequence of efficient spectrum use is 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 bytes.

Header compression methods maintains a context, which is essentially the uncompressed version of the last header sent over the link, at both compressor and decompressor. Compression and decompression is 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 in 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 an ideal header compression scheme 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 gets 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 CRTP with TWICE lost around two times as many packets than the ideal header compression scheme. 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 occur 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 for CRTP:s bad performance over cellular links is the long link roundtrip time, during which many packets are lost when the context is out of sync. That 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, they 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 has 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 do not state any exact details about how header compression is performed, various header formats or so on, that 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 only things exactly stated by the framework are the rules regarding negotiation of compression profiles.

5.2. General ROCCO principles

ROCCO header compression is based on the principle with de-compressor context repair attempts relying on a header compression CRC included in compressed headers. Profiles will define various packet types and all of them MUST NOT 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 different 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, considering the fact that 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 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 models 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 that have occurred in the packet stream.

5.4. Header compression profiles

The details on how a packet stream is to be compressed and decompressed is 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 subsequent chapter.

One way to select a profile can be to have a common channel over which a general-purpose header compression scheme, perhaps CRTP, is used. When its found that a packet stream suits a specific header compression profile, 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 length of and what the Header Compression CRC covers for all packet types.
- the information needed in the contexts for compression and decompression, which will include history information and properties of the packet stream.
- procedures for compression and decompression.
- how compression is initiated (which packets 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, the compressor must be able to send a message to the decompressor about what header compression profile to use. This message consist of a 16 bit profile identifier, and underlying link layers MUST provide a way to transfer this message.

5.6. Link-layer requirements

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

Framing

Framing is the most important link layer functionality making it possible to separate different packets.

Length

Most link layers can indicate the length of the packet and that 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 do 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 by 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 which values defines a packet stream. They are in general handled as STATIC.
STATIC-KNOWN	These STATIC fields are expected to have well known values and are therefore not needed to be communicated at all.
CHANGING	These fields are expected to vary in some way, either randomly, within a limited scope, with constant values, or by other means.

All not changing 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 reflect 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 : size of field classes

The information carried in the STATIC and STATIC-DEF fields have to be transferred once, and every header compression profile MUST specify a way for doing this. 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. 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.

7. Header compression profiles for IP-telephony packet streams

This section is an example showing how the framework outlined in 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 up to three 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 specialized on 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 classified as 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	These are fields that changes their values occasionally and then keeps their new values.
ALTERNATING	These fields have a few different values which they are alternating between.
IRREGULAR	These are finally the fields for which no useful change pattern can be identified.

To further expand the classification possibilities without making it more complex, 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, something should finally 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 behaviors, it could be known that changes use to be within a LIMITED scope

compared to the maximal change for the field. For others, 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 discusses the various header fields in detail. Note that table 7.1 and the discussions below does 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 the same 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. Thereby, the delta value of the field is constant and well known a priori. When RTP is used on top of UDP and IP, this means that the IP ID value would follow the RTP sequence number. This assignment policy is the most desirable for header compression purposes but usage of it is not very common. The reason for that 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 used 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 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 that sends the ID field 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 senders assignment policy could therefore be useful when choosing between the two solutions above. Finally, profiles could also for IPv4 streams 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 make 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 are 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.

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 will be 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 if CSRC items present in the CSRC list. This number is expected to be almost constant on a packet-to-packet basis and change with small values. 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 is expected to be a rare case. 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 with 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 with 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 have a relatively limited scope.

7.2.10. RTP Contributing Sources (CSRC)

The participants of a session, which are identified by the CSRC fields, are expected to be almost the same on a packet-to-packet basis with relatively 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 requirements and capabilities of each profile in combination with information about which mechanisms that are used to implement the desired behaviors.

7.3.1. List of defined profiles

All defined profiles are listed in Table 7.2 together with their characteristics and pointers to implementation details that may differ from profile to profile. The meaning of the columns and possible values for each of them are listed below the table.

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

Table 7.2 : List of defined profiles

The first six columns states requirements 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.

IPv

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

Str

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 UPC checksum (E)nabled or D(isabled).

Id

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

(S)EQUENTIAL	These profiles can not 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.
(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 with two octets.

MHS

Minimal Header Size for the profile.

The next five columns indicates how each profile is implemented. This includes header formats for STATIC (STA, chapter 7.6.1), CONTEXT_UPDATE (CUP, chapter 7.6.2) and COMPRESSED (COM, chapter 7.6.3) packets, and also what EXTENSION (EXT, chapter 7.6.4) formats that are used with the COMPRESSED packets and the interpretation of the Code-field (CFI, chapter 7.6.5).

7.3.2. Additional, common profile characteristics

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

Link layer requirements

Except for the general link-layer requirements (framing, length & profile negotiation) stated in chapter 5.5, these profiles require also 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.

Packet stream characteristics

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

Pre link characteristics

Up to three consecutive packet losses before the header compression point are for most of these profiles handled without introducing additional header overhead. Packet reordering on pre links is expected to be very uncommon.

Link Characteristics

The link over which header compression is performed is expected to have a loss characteristic that very seldom leads to loss of more than three consecutive packets. The round-trip time is expected to be between 100 and 200 milliseconds.

7.4. Packet types and Code-field usage

The profiles defined in this document makes use of four different packet types; STATIC, CONTEXT_UPDATE, COMPRESSED and CONTEXT_REQUEST. Detailed descriptions of packet formats are given in chapter 7.6.

To identify packet types, four bit patterns for the initial five bits of the first octet are reserved. These patterns are:

STATIC	00000
CONTEXT_UPDATE	00010
CONTEXT_REQUEST	000-1

All other bit-patterns indicates a COMPRESSED packet format and the meaning of those patterns are defined together with the specifications of the packet formats. These five bits are hereafter referred to as the Code-field in COMPRESSED headers.

7.5. Sequence number encoding

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, 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 task is then to communicate RTP sequence number information in some way. If delta encoding is used, it has to tolerate up to two or three consecutive lost packets on the link between compressor and decompressor. This chapter describes how to do this by using a partially redundant encoding together with special reconstruction attempts.

7.5.1. Sequence number changes to handle

The source increases the RTP sequence number with one for each packet sent. However, due to losses and reordering the changes seen by the compressor will vary. If we consider our scenario in Figure 1.1 where there are cellular links at both ends of the path, it seems reasonable to set the same algorithm requirements for loss on both cellular links. This implies that three consecutive packets could be lost. Loss on the wired portion of the path is assumed to be insignificant in comparison, so the maximal number of lost packets to be handled efficiently is three also before the header compression point. An example of possible sequence number changes seen by the compressor would then be as shown in Figure 7.1.

From sender :	1	2	3	4	5	6	7	8	9	10
Lost on path :		X	X				X	X	X	
Received by HC :	1			4	5					10
Sequence delta :	-			3	1	1				4

Figure 7.1: possible sequence number changes.

Possible sequence delta values are then: 1, 2, 3 or 4

Packet reordering is uncommon, and should therefore not be handled by the efficient encoding.

The sequence deltas to handle are thus: 1, 2, 3 and 4. When packets are duplicated in the network, the delta can be 0 (zero). We do not deal with such deltas because duplicated packets MUST NOT be sent over the cellular link; they MUST be discarded.

7.5.2 Compression of sequence number values

The sequence number encoding is based on two values called Individual Delta (ID) and Accumulated Delta (AD). These two are further merged to one value, called the Encoded Delta (ED), which has to be encoded and sent in compressed headers. The values are calculated as follows.

ID: The Individual Delta is the sequence number delta from the previous packet sent from the compressor. The value can be 1, 2, 3 or 4.

AD: The Accumulated Delta is a sum of the Individual Delta values of the three previous packets sent from the compressor. The possible values of AD will be 3, 4, 5, 6, 7, 8, 9, 10, 11 and 12.

ED: ID and AD are mapped to a single value, ED. How this is done and encoded for different compressed headers are described in chapter 7.6.5 and Appendix A.

The purpose of transmitting the encoded delta values is to make it possible to recover from packet loss between compressor and decompressor.

7.5.3. Decompression of RTP sequence number values

The decompressor makes attempts to decompress based on assumptions on the number of lost packets between compressor and decompressor since the last packet received and successfully decompressed. For each attempt the header is reconstructed according to the assumed number of lost packets, and the correctness is verified with the Header Compression CRC. To reconstruct the header the ID and AD values are used to calculate the sequence number corresponding to the attempt. The decompressor needs a short history of previous IDs, ADs and RTP sequence numbers to do this. The details of the algorithm are explained in the following:

Let the decompressor keep an imaginary counter, S, of the packets received and let N be the number of the current packet. S(N) is the calculated RTP sequence number for packet N. The attempts are made in the following order:

1 - No loss: S(N) = S(N-1)+ID(N)

2 - One lost: S(N) = S(N-1)+ID(N)+AD(N)-ID(N-1)-ID(N-2)

3 - Two lost: S(N) = S(N-1)+ID(N)+AD(N)-ID(N-1)

4 - Three lost: S(N) = S(N-1)+ID(N)+AD(N)

5 - Special: S(N) = S(N-2)+ID(N)+AD(N)

If attempt 5 fails, the decompression is not guaranteed to succeed. A decompressor MAY make additional repair attempts. A CONTEXT_REQUEST MUST be sent to the compressor to request an update to repair the context if the repair fails.

Note that the decompressor MUST calculate the ID value for all packets, also for the ones sent with UPDATE headers.

7.6. Header formats

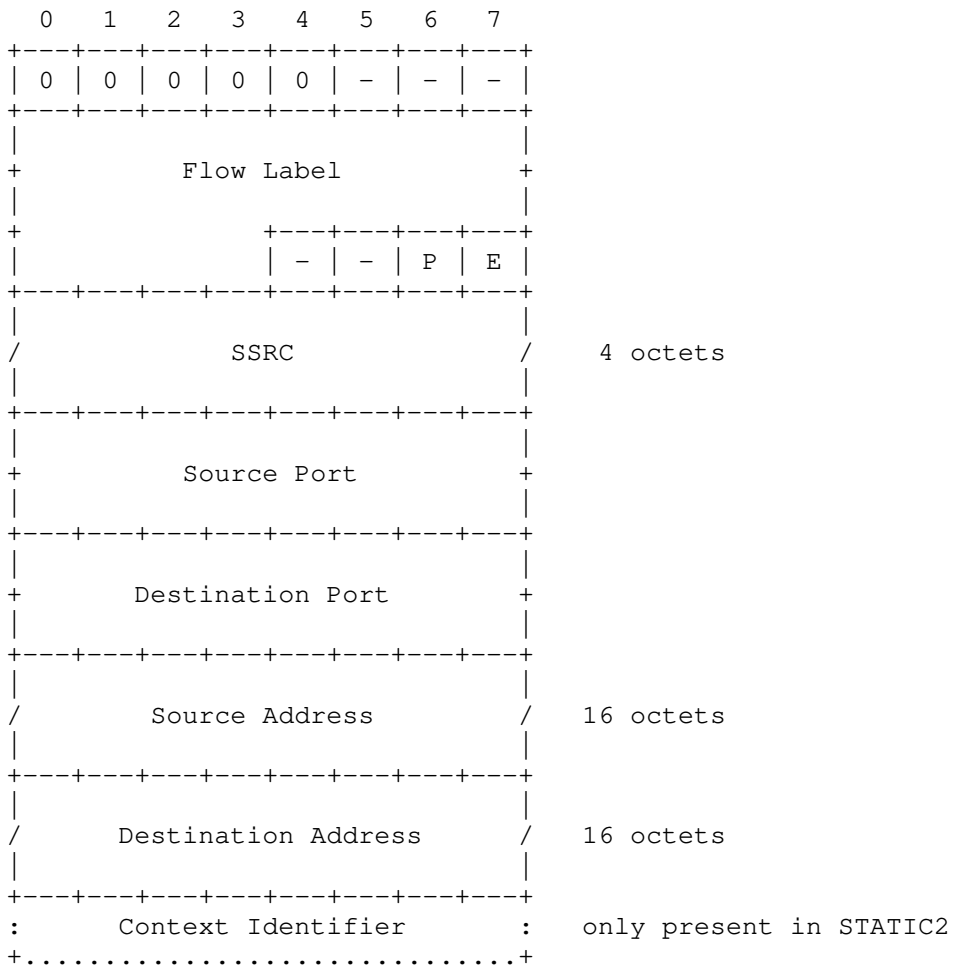
This section defines the header formats of the four packet types STATIC, CONTEXT_UPDATE, COMPRESSED and CONTEXT_REQUEST together with descriptions of when and how to use them. Subsections are also dedicated to the EXTENSION formats of COMPRESSED headers and to the interpretation of the Code-field for different profiles.

7.6.1. Static information packet, initialization

The STATIC packet type is a packet containing no payload but only the header fields that are expected to be constant within 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 decompressor (see 7.6.6 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.

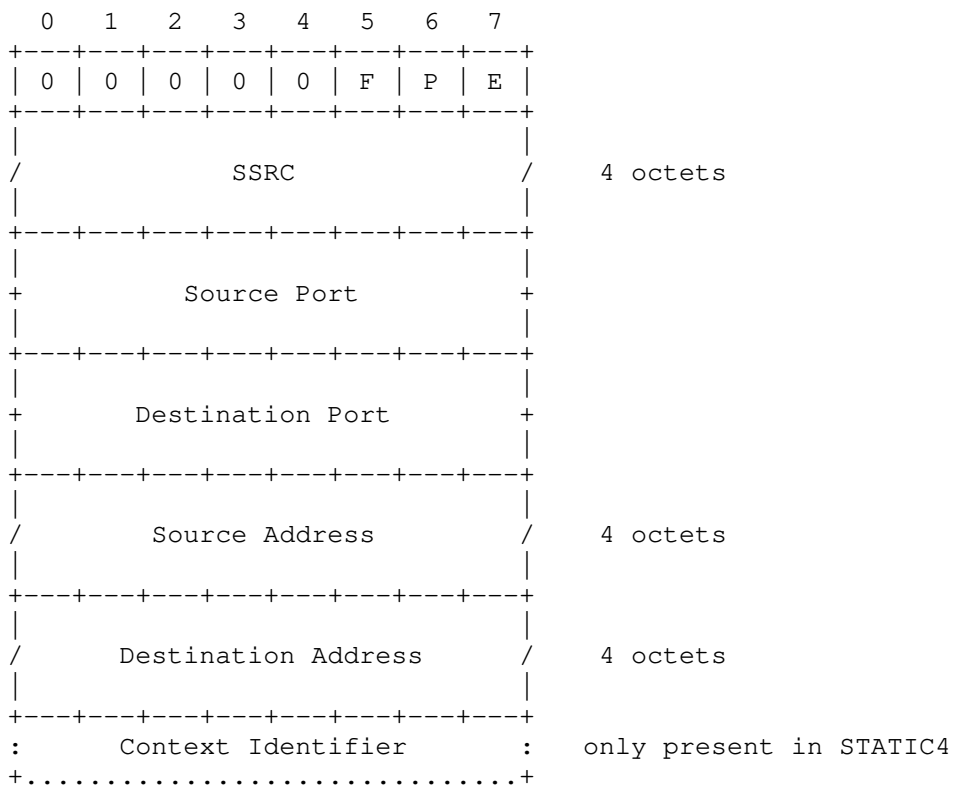
IPv6 (44-45 octets)

Defines packet types: STATIC1, STATIC2



IPv4 (17-18 octets)

Defines packet types: STATIC3, STATIC4



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

Only one STATIC packet is sent at each occasion. If the decompressor receives compressed headers or updates without having received a STATIC packet, the decompressor MUST request a STATIC packet.

7.6.2. Context update packet

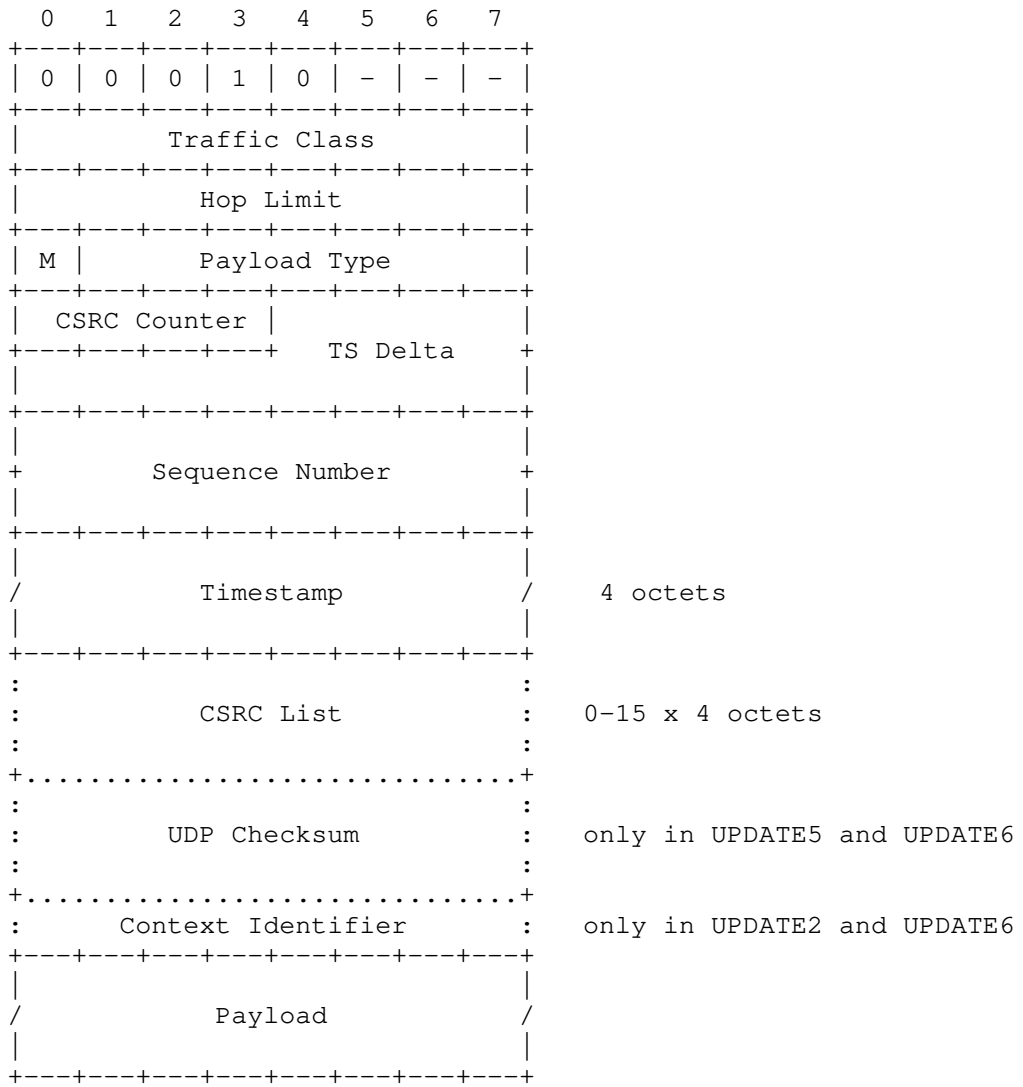
The CONTEXT_UPDATE packet type has a header containing all changing header fields in their original, uncompressed form, and carries a payload just as ordinary COMPRESSED packets. Four CONTEXT_UPDATE packets MUST be sent after the initial STATIC packet to set up the decompressor context for the first time. In addition, this update packet type MUST be used whenever the decompressor requests a context update, and when the header fields change in a way that cannot be encoded in COMPRESSED packets.

All fields except for the initial five bits, the padding (-) and the Timestamp Delta are ordinary IP, UDP and RTP fields. The Timestamp Delta is the current delta between RTP timestamps in consecutive RTP packets. Initially this value is 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 CONTEXT_UPDATE packet types.

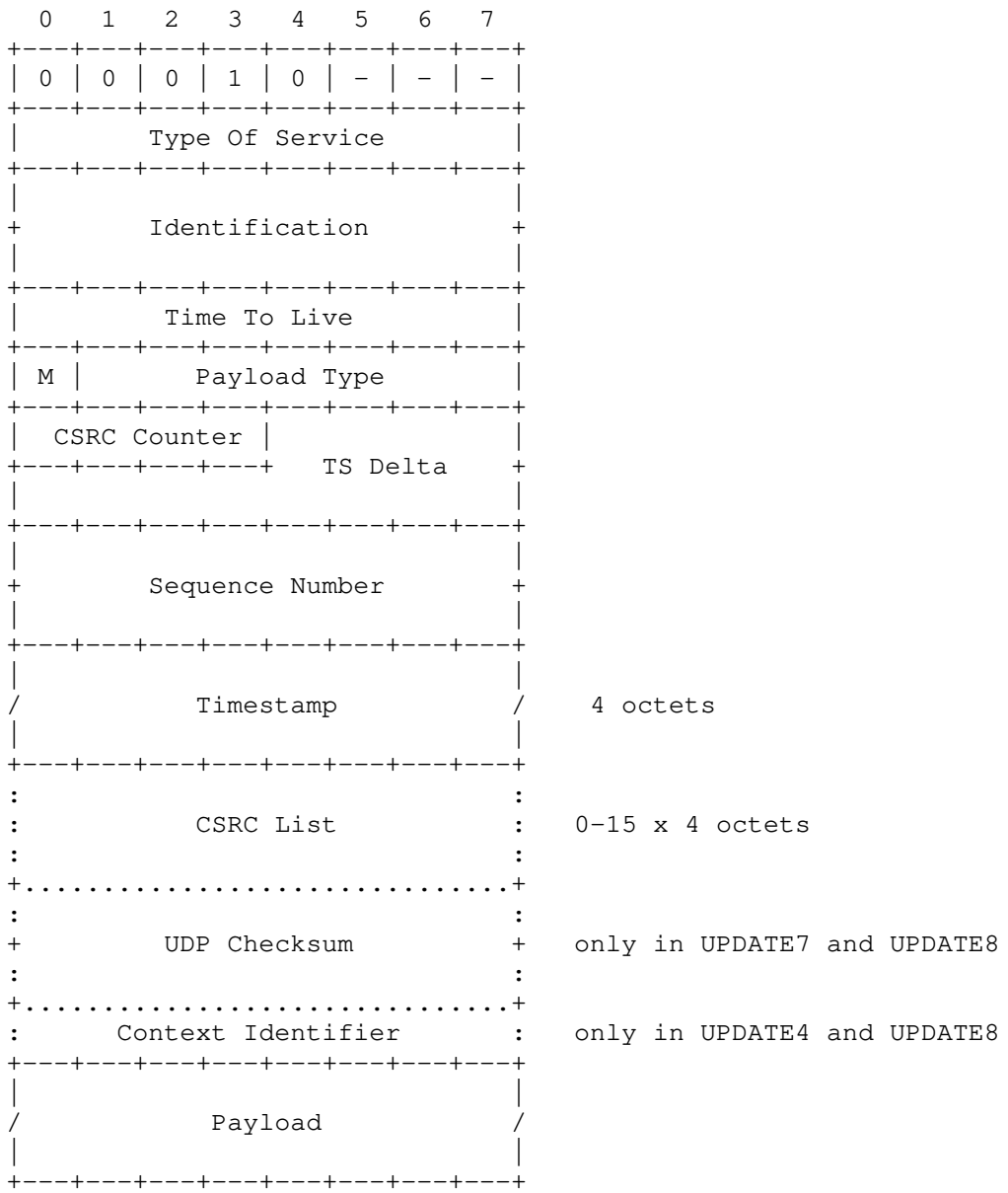
IPv6 (12-15 octets + CSRC List of 0-60 octets)

Defines packet types: UPDATE1, UPDATE2, UPDATE5, UPDATE6



IPv4 (14-17 octets + CSRC List of 0-60 octets)

Defines packet types: UPDATE3, UPDATE4, UPDATE7, UPDATE8

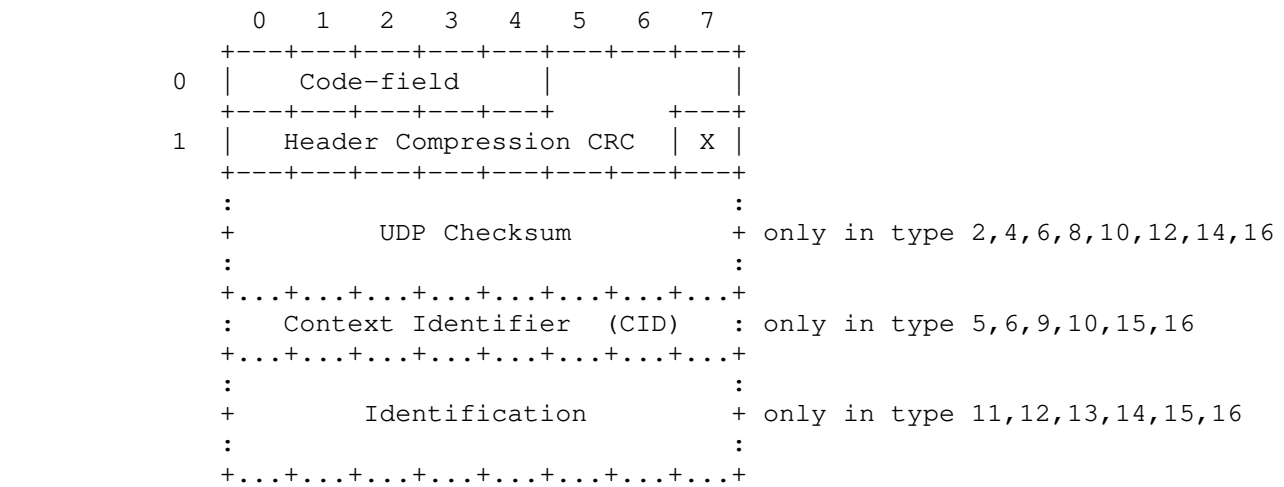


Each time a CONTEXT_UPDATE packet is sent, the three subsequent packets MUST also be CONTEXT_UPDATE packets. This ensures that the update will succeed even when three consecutive packets are lost.

7.6.3. Compressed packets

The COMPRESSED packet type is the most commonly used one 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. 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. Neither the Code-field nor the extension bit is included in the checksum computation. The CRC polynomial to be used is:

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

The interpretations of the Code-field for different profiles are given in section 7.7.5.

7.6.4. Extensions to compressed headers

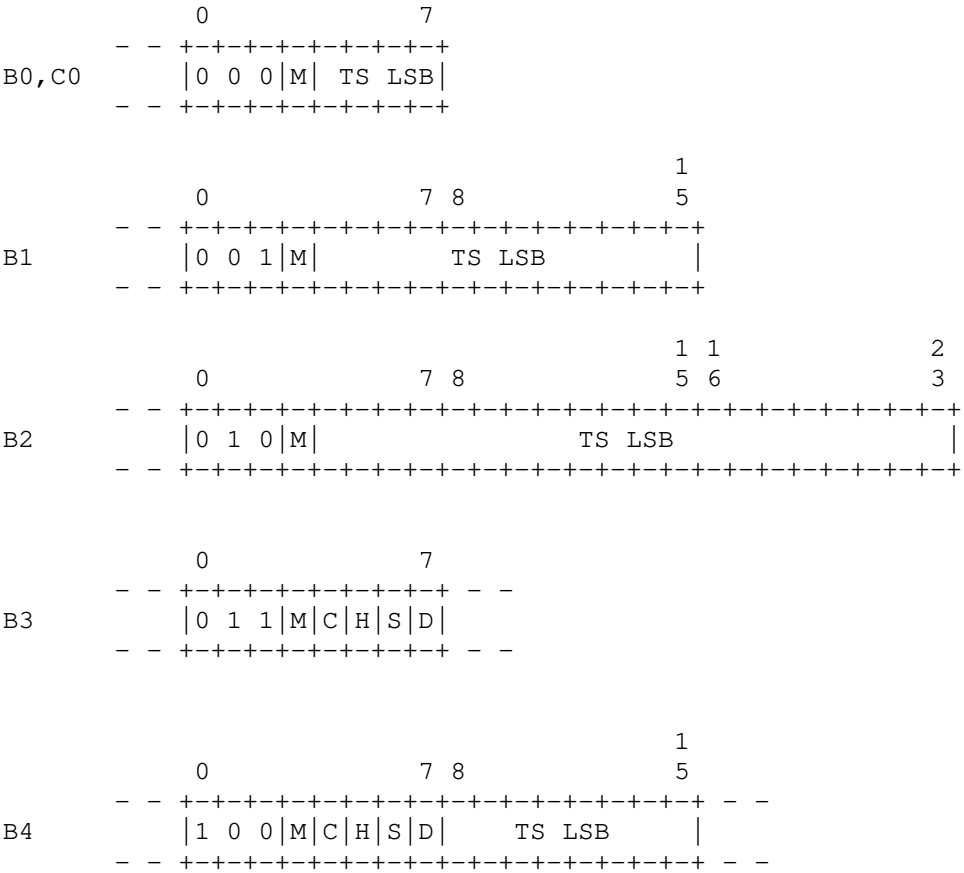
Less regular changes of the header fields require an extension in addition to the base header. When there is an extension present in the COMPRESSED packet, it is indicated by the extension bit (X) being set. Extensions are of variable size depending on the information needed to be transmitted. However, the first tree extension bits are for all extension formats used as an extension Type field. The extension can carry an M bit, a TS LSB field, a SEQ field, an ID 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

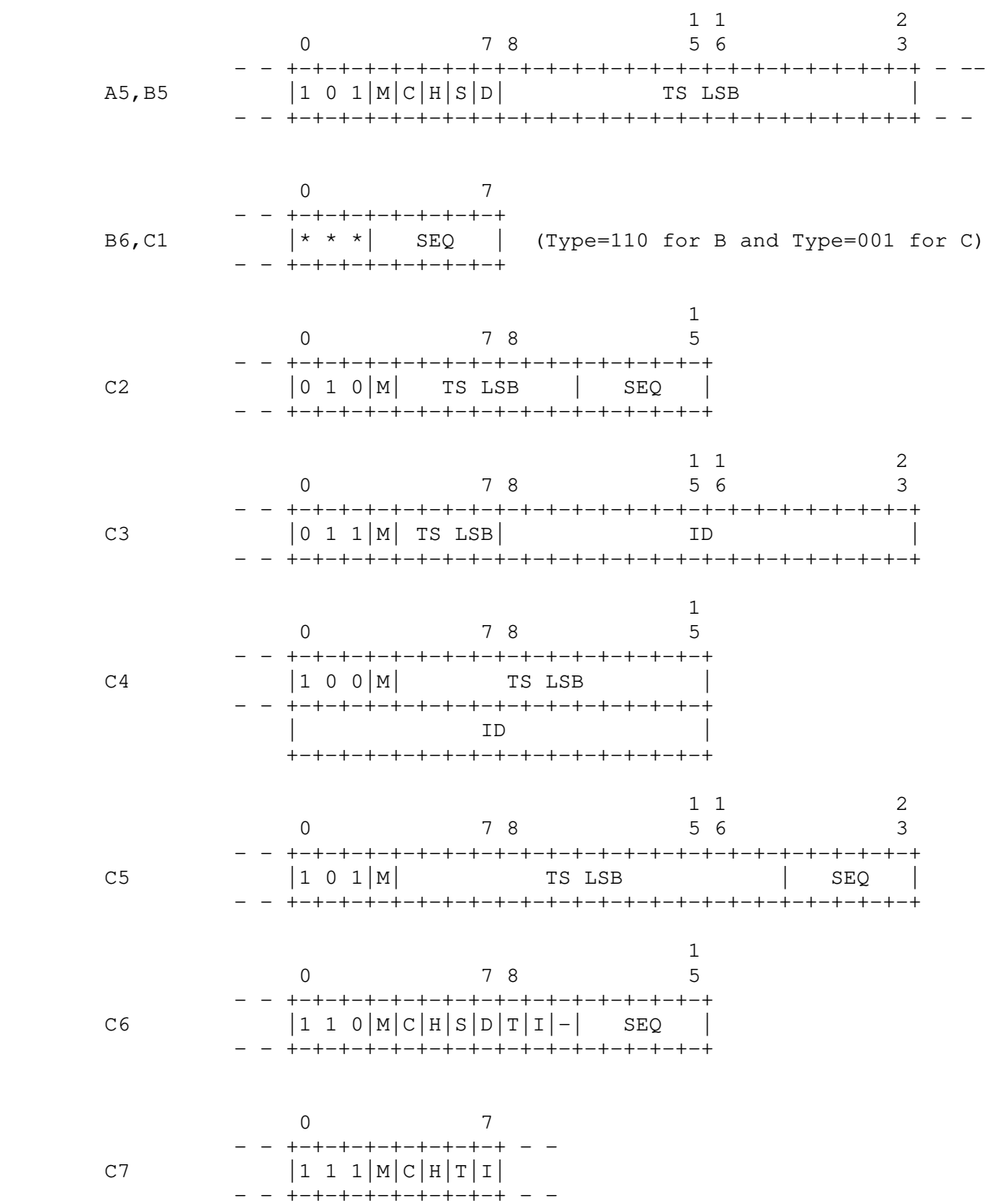
timestamp value (the most significant bits are then expected to be unchanged since previous packets). The SEQ field is used for sequence number encoding and the ID is the ordinary IP Identification field. Various bit mask patterns are possible and can consist of S,H,C,D,T and I. The interpretations of these bits are given in the end of this chapter.

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

For the profiles defined in this document, three different extension sets are used, called A, B and C. Set A and B are almost identical with the only difference that B make use of one additional extension type compared to A. Set C handles much more functionality and it differs therefore more. All possible extensions are shown below with indications of which sets and types the extensions corresponds 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:




A5,B5
1 0 1 M C H S D
TS LSB
B6,C1
* * *
SEQ
(Type=110 for B and Type=001 for C)
C2
0 1 0 M
TS LSB
SEQ
C3
0 1 1 M
TS LSB
ID
C4
1 0 0 M
TS LSB
ID
C5
1 0 1 M
TS LSB
SEQ
C6
1 1 0 M C H S D T I -
SEQ
C7
1 1 1 M C H T I

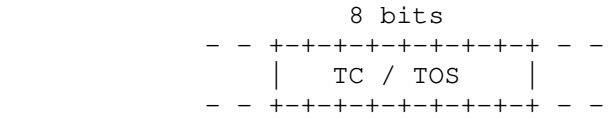
A bit mask indicating additional fields could include bits with the following meaning:

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

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.

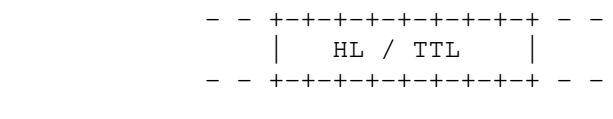
C - Traffic Class / Type Of Service

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



H - Hop Limit / Time To Live

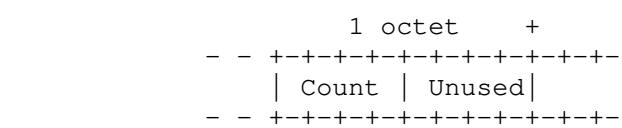
The field contains the value of the original IP header field.



S - Contributing Sources

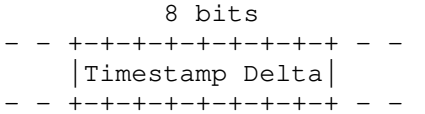
The CSRC field is built up of:

- 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)



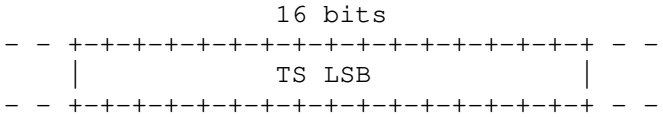
D - Timestamp Delta

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 instead the number of samples divided by 8. Thus, only Timestamp Delta values evenly divisible by 8 can be communicated in a Timestamp Delta field in 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 CONTEXT_UPDATE packet.



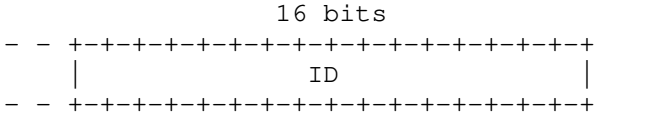
T - Timestamp LSB

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

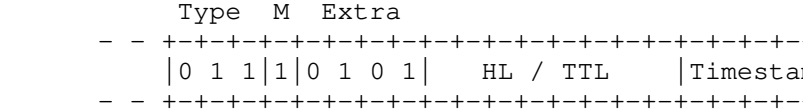


I - Identification

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



An example where the HL/TTL and the Timestamp Delta fields are present in a type A3 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 MUST also be sent in three subsequent packets (either as Extensions or CONTEXT_UPDATEs) to satisfy the three-packet-loss requirement.

7.6.5. Interpretations of the Code-field

The usage of the Code-field in COMPRESSED headers (except for the packet type identification using four code points) differs from profile to profile. The field is used in three different ways for the profiles defined in this document:

S - Sequence encoding

The remaining 28 code points are used for robust encoding of sequence numbers as described in chapter 7.5 and Appendix B.

C - Context Identification (CID)

The code points are used to separate up to 28 different packet streams. When used this way, the sequence encoding MUST be done in extension headers. The encoding is performed using the same principles as for S above, but using the SEQ field of an extension instead. However, as long as the sequence values are the ordinary ones (ID=1, AD=3) the sequence information MAY be omitted, saving one header octet. If the decompressor receives a packet without sequence information, it knows that the values are the most common ones.

S/I - Sequence encoding OR IP Identification LSB

For profiles using this Code-field interpretation, the field could be used both for sequence encoding and for transmission of the least significant bits for the IP Identification field. The standard usage MUST be for IP Identification because the sequence encoding has "most common values" (ID=1, AD=3), which requires no sequence code information at all to be sent for such packets. Sequence information MUST only be sent if it differs from this standard case, and it could then either be sent in the Code-field or in the SEQ field of an extension header. If the IP identification has changed too much to be sent in the Code-field, it MUST be sent in an extension instead and for those cases, the sequence encoding MUST be sent in the Code-field. Otherwise the sequence code MUST be sent in an extension, when needed.

7.6.6. Context request packets

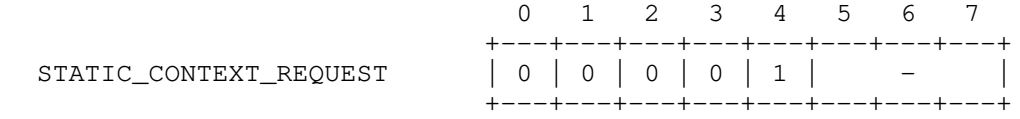
CONTEXT_REQUEST packets are used by the decompressor to request context updates from the compressor. This is done when the context of the decompressor is not valid or not in sync and decompression therefore is impossible. The main reasons for an invalid context are:

- The context initialization has failed.

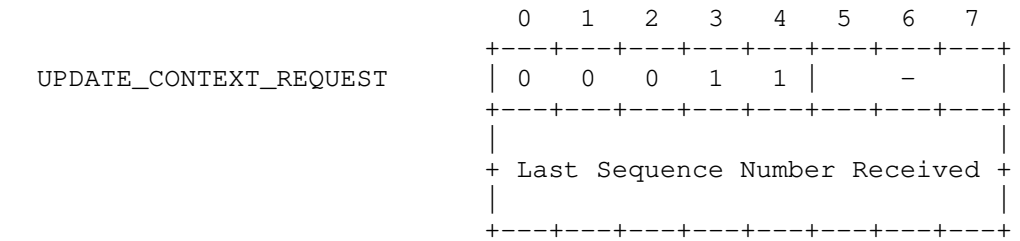
- The context has been brought out of sync due to errors on the link and the context repair mechanisms have failed.

To set up the context in the first place, a STATIC packet MUST arrive to the decompressor to install the static parts of the context. Then a CONTEXT_UPDATE packet is REQUIRED to install the CHANGING parts of the header plus packet stream related information. COMPRESSED packets can be handled only after both these packets have been received.

There are two kinds of CONTEXT_REQUEST packets. The first kind is used to request a new STATIC packet and is normally sent only if the first STATIC packet was lost or damaged and other packets arrive. The format of this STATIC_CONTEXT_REQUEST packet is shown below.



The other kind of CONTEXT_REQUEST requests a CONTEXT_UPDATE packet and is sent whenever COMPRESSED packets arrives but the decompressor fails to decompress them. The format of this packet is shown below.



7.7. Context update procedures

How to send and respond to CONTEXT_REQUEST packets are 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. The sequence number of the last packet header correctly reconstructed by the decompressor is provided in each update request, and could be used by the compressor when deciding how to respond to the request.

8. Simulated performance results

To evaluate the performance of ROCCO and the IP telephony profile, we have simulated three header compression schemes; CRTP [CRTP], ROCCO, and the Ideal header compression scheme defined in chapter 2. the ROCCO profile used is number 7 of table 7.2. Sections 8.1 to 8.5 provides the parameters used in the simulations. Sections 8.6 and 8.7 show the results.

8.1. 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 being compressed over the last cellular link using one of the three header compression schemes. The simulated scenario is shown in Figure 8.1.

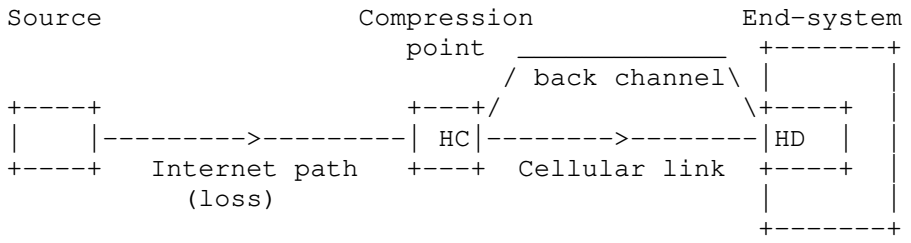


Figure 8.1 : Simulated scenario.

8.2. 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 bytes).

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.

8.3. Influence of pre-HC links

A worst case scenario is considered. The packet loss rate is uniformly distributed with a probability of 1 %. First degree reordering is also uniformly distributed with a probability of 1 %. No higher reordering degree is considered. The purpose of using high error probabilities is to test the capabilities of the schemes also under tough conditions. The speech quality would suffer at such error rates.

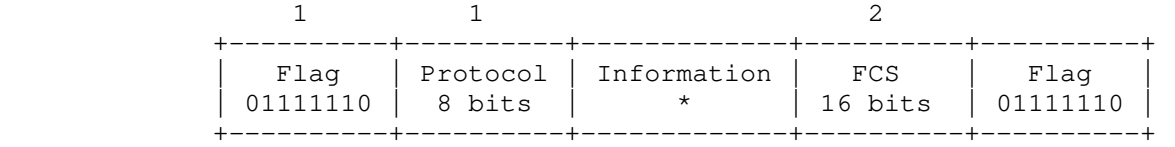
8.4. Used link layers

Two link layers are considered in the algorithm evaluation, as in [CRTPC]. One is PPP in HDLC-like framing [HDLC], which has a 16-bit CRC covering the entire frame. This implies that all damaged frames are discarded at the link layer.

The second link layer used is an imaginary framing scheme called LLPC, Link Layer with Partial Checksum. As the name implies, the CRC does not cover the whole frame. The payload (speech data) is not covered by the CRC, which fits well with speech coders that can conceal some damage. The partial checksum will increase the number of headers seen by the decompressor and hence improve header compression performance.

8.4.1 PPP in HDLC-like framing (HDLC)

PPP typically uses HDLC-like framing [HDLC]. With a 16-bit checksum and compressed Address and Control fields the frames have the following format.

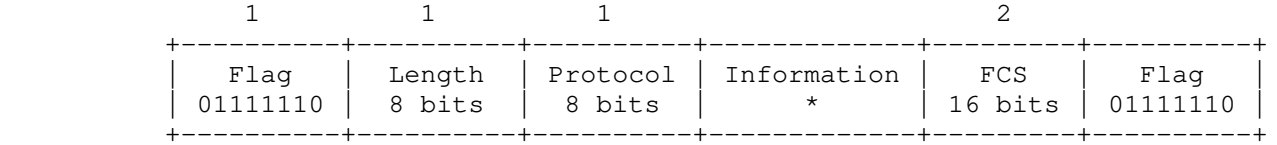


The Flag only occurs once between frames if they are sent back-to-back, so the amortized framing overhead is 4 octets per frame. The checksum (FCS) is calculated over the Protocol field and the Information field (payload), but not the Flags or the checksum itself.

Any errors anywhere in the frame will cause the FCS to fail. The frame will then be discarded.

8.4.2 Link-layer with partial checksum (LLPC)

This is an imaginary framing scheme derived from the HDLC-format in 8.4.1 by adding a one-octet Length field.



The Length field indicates how many octets of the payload that are covered by the FCS. It can have values from 0 to 255. The FCS covers

the Length and Protocol field plus as many octets in the beginning of the Information field as indicated by the Length field. The value of the Length field must not make the FCS extend over the FCS field. When sending a FULL_HEADER packet, the Length field would have the value 60 for IPv6 (40 for IPv4), since it should protect the IP, UDP, and RTP headers. When sending a minimal COMPRESSED_RTP or ROCCO packet, the Length field would have the value 2. The amortized framing overhead for LLPC is 5 octets per frame.

Any errors in the Flag, Length, Protocol, FCS, or the initial Length octets of the Information field will cause the FCS to fail. The frame will then be discarded. Errors in the Information field after the first Length octets will not affect the FCS and will not cause the frame to be discarded.

8.5. 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 is thus the BER after channel coding.
The back channel used in our simulations never damages the CONTEXT_REQUEST messages. The RTT is set to 120 ms.

8.6. Compression performance

Figure 8.2 shows the average header size plotted against BERs for the three header compression schemes using HDLC. For BERs below 1e-4, both CRTP and ROCCO give an average header size of just above 2 octets, 2.25 for CRTP and 2.15 for ROCCO. The average header size for CRTP starts to increase when the BER gets higher than 1e-4 and at BER 1e-3 it is 3.35 octets. For ROCCO the average header size is constantly 2.15 octets up to BER 1e-3. For higher BERs it increases slightly to reach 2.75 octets at BER 1e-2, where CRTP has an average header size of 5.20 octets.

Figure 8.3 shows the same plots for LLPC. The average header size for ROCCO remains constant at 2.15 octets up to a BER of 5e-3. CRTP header size on the other hand starts to increase at BER 3e-4. The average header size for CRTP is 2.70 octets at BER 1e-3 compared to 2.15 for ROCCO. At BER 1e-2, CRTP gives an average header size of 4.20 octets, and ROCCO 2.25 octets. The difference between CRTP and ROCCO is mainly that the latter tolerates losing up to 2 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 loosing a significantly lower number of packets.

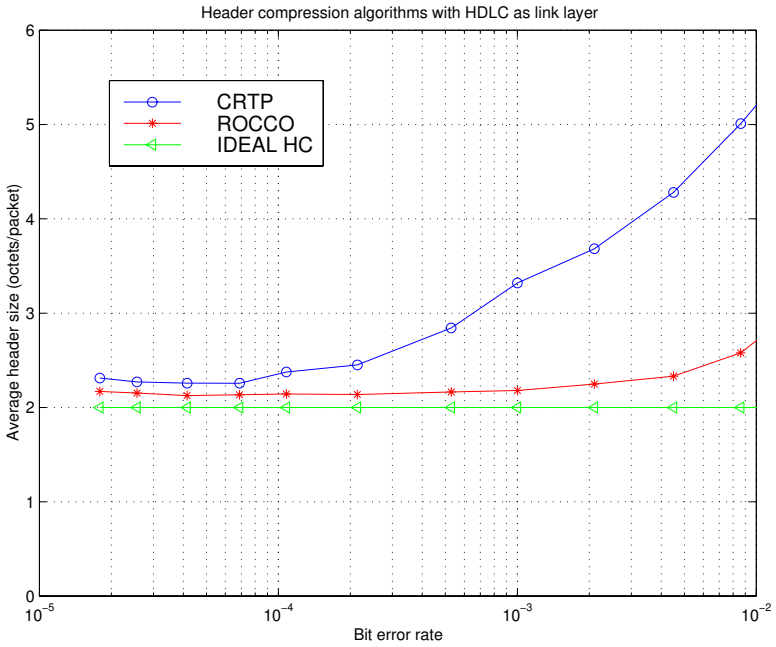


Figure 8.2 : Average header sizes with HDLC framing

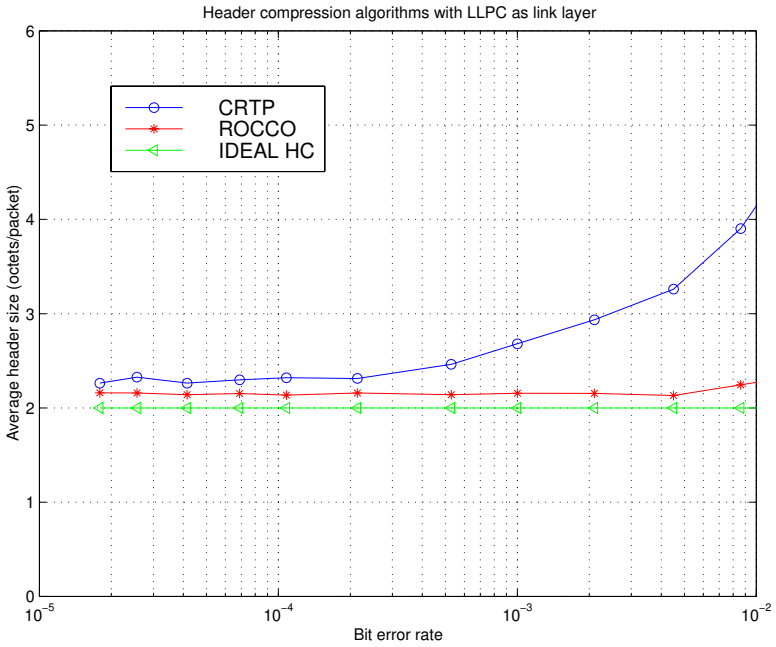


Figure 8.3 : Average header sizes with LLPC framing

8.7. 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 for LLPC is not regarded as lost as long as it is deemed ok by the header compression scheme.

In Figure 8.4 the FER for HDLC is shown for the three header compression schemes. At BER 1e-4 we have for CRTP 0.78 % FER, for ROCCO 0.18 % and ideal HC gives 0.14 % FER. When increasing the BER to 1e-3, CRTP gives 16.56 % FER, ROCCO 3.69 % and ideal HC 3.36 %.

Figure 8.5 shows the FER for LLPC. ROCCO and ideal HC have a FER of 0.98 % and 0.69 %, respectively, while CRTP has a FER of 5.27 %. Given the performance of the ideal scheme, it is clear that most of the CRTP loss is due to context damage. ROCCO is close to the ideal scheme until the BER gets so high that more than 2 consecutive packets often are lost on the link.

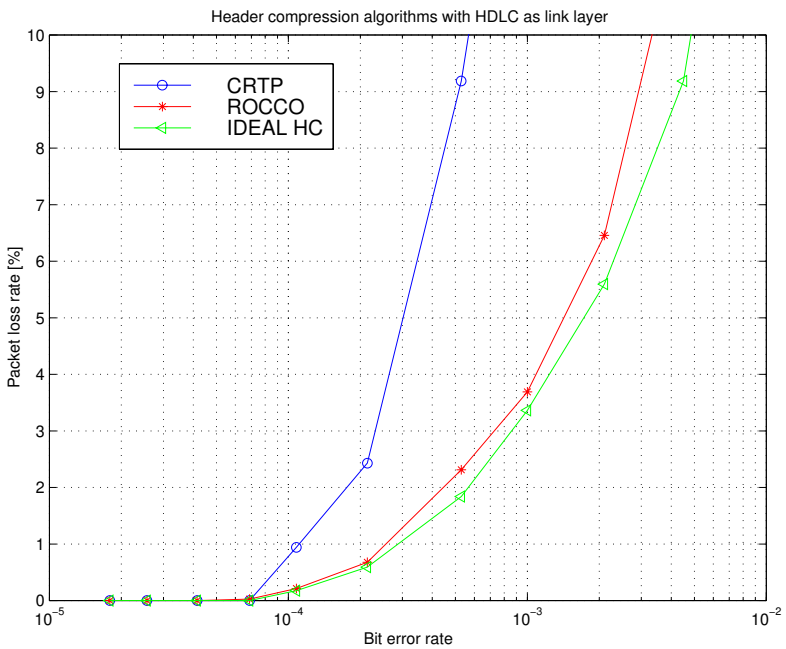


Figure 8.4 : Packet loss rate versus BER with HDLC framing

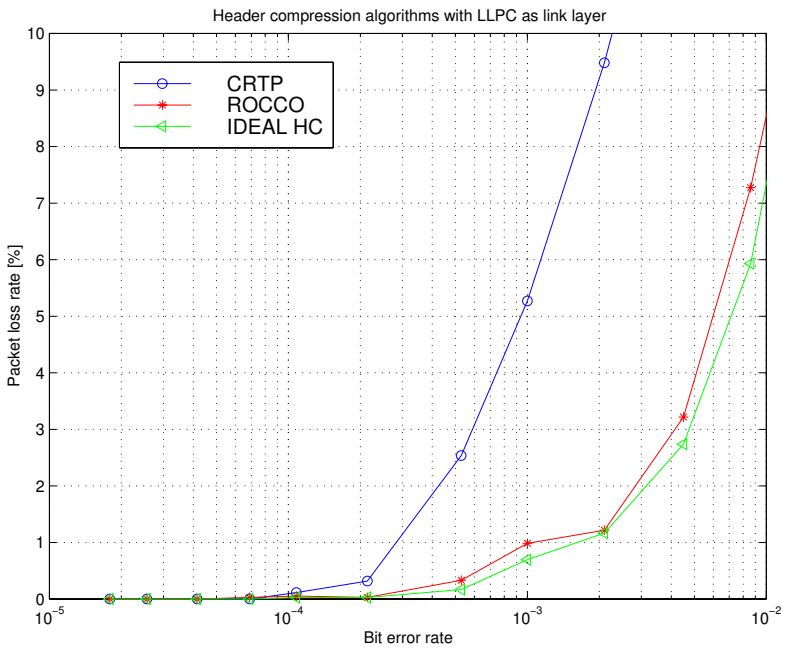


Figure 8.5 : Packet loss rate versus BER with LLPC framing

Figures 8.6 and 8.7 show the loss pattern for CRTP and ROCCO with HDLC and LLPC at BER 1e-3. It is evident from these figures that the majority of loss events with CRTP are such that around 7 consecutive frames are lost. The link roundtrip time in these simulation were 120 ms and the packet rate 50 packets per second, which means that a single discarded frame causes 6 additional frames to be lost due to context damage. For ROCCO, almost all loss events include 1 or 2 lost frames, which means that it does not suffer from context damage.

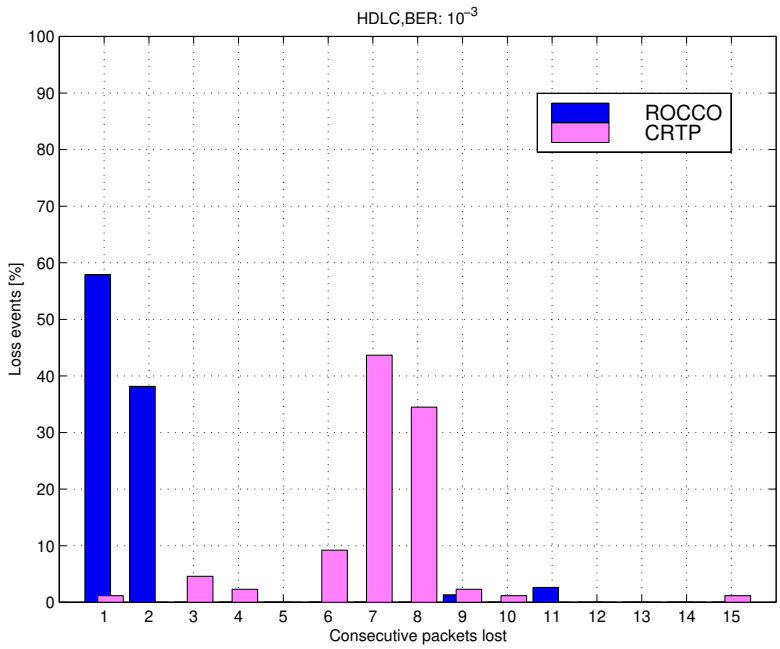


Figure 8.6 : Packet loss patterns for CRTP and ROCCO with HDLC

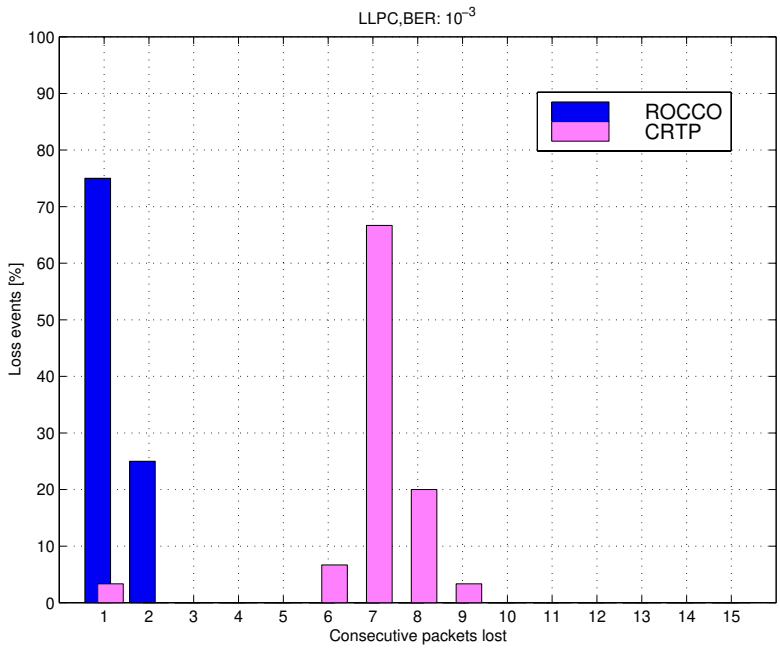


Figure 8.7 : Packet loss patterns for CRTP and ROCCO with LLPC

8.8. CRC strength considerations

The 10 bits of CRC is 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 argues therefore that 10 bits of CRC is enough to verify the correctness of the guessed header.

9. Implementation status

The ROCCO algorithm, as stated in draft version 01, has been implemented in a testbed environment for real-time IP traffic over wireless channels. In the testbed it is possible to listen to the effects of header compression in conjunction with packet losses. The current implemented profile is 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. The implementation has made some impact on the ROCCO protocol, realized in this document.

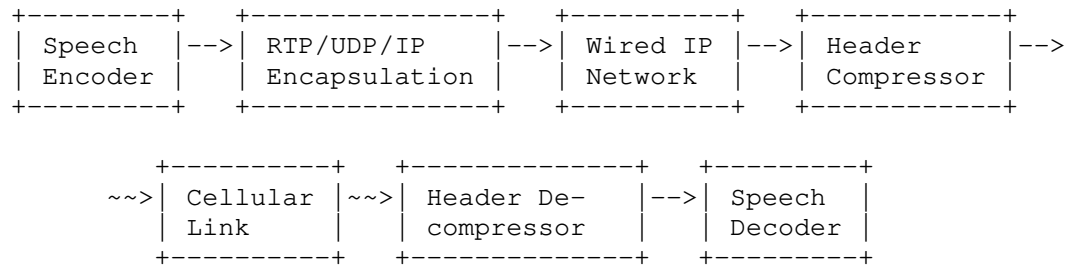


Figure 10.1 : Block diagram of testbed environment

Continuously updated information about implementation status can be obtained from the ROCCO-Homepage:
http://www.ludd.luth.se/users/larsman/rocco/

10. Security considerations

Because encryption eliminates the redundancy that header compression schemes try to exploit, there is some inducement to forego encryption in order to achieve operation over low-bandwidth links. However, for those cases where encryption of data and not headers is satisfactory, 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 allow header compression to still 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 check-sums. 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 header not matching the original ones without this being noticed.

Denial-of-service attacks are possible if an intruder can introduce (for example) bogus STATIC, CONTEXT_UPDATE or CONTEXT_REQUEST packets onto the link and thereby cause compression efficiency to be reduced. However, an intruder having the ability to inject arbitrary packets at the link-layer in this manner raises additional security issues that dwarf those related to the use of header compression.

11. Conclusions

This document specifies a framework for header compression over lossy links based on checksums and local repairs: ROCCO. It also defines compression profiles suited for compression of RTP/UDP/IP headers in IP telephony packet streams.

The compression profiles are evaluated by simulations over cellular links with high but not unrealistic error rates. The performance is excellent and very close to the performance of an ideal header compression scheme. Compared to CRTP, the packet loss rate of ROCCO is around 4-6 times less over links with high error rates. Moreover, it does not, as does CRTP, introduce loss events involving many packets.

CRTP on the other hand performs well when error rates are low. It is also general in the sense that it is not very dependent on the properties of the RTP streams it compresses. CRTP is a good candidate for channels with low error rates and in cases when many different RTP streams are intermixed.

12. Acknowledgements

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

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

13. Intellectual property considerations

Ericsson has filed patent applications that might possibly have technical relations to this contribution. For an Ericsson statement regarding these applications and possible patents affecting this proposal, see the IETF IPR statements page:
http://www.ietf.org/ipr.html

14. References

[UDP] Jon Postel, "User Datagram Protocol", RFC 761, 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), June 1999. <draft-degermark-crtp-cellular-00.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>

[WCDMA] "Procedure for Evaluation of Transmission Technologies for FPLMTS", ITU-R TG8-1,8-1/TEMP/233-E, September 1995.

15. Authors' addresses

Lars-Erik Jonsson	Tel: +46 920 20 21 07
Ericsson Erisoft AB	Fax: +46 920 20 20 99
Box 920	Mobile: +46 70 365 20 58
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 70 378 04 73
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 kind of packet streams respectively.

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 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.

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, being one of the field 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 well 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, but instead regenerate it 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 exist only one working RTP version and that is version 2. The field is therefore classified as STATIC-KNOWN.

Padding

This field is depending 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 an 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 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. Mapping of ID and AD values (According to chapter 7.5.2 and 7.6.5)

Both when encoded in the Code-field of compressed base headers and when encoded in extensions, there are five bits reserved for this encoding. For the Code-field, four bit patterns (000--) are reserved, which leaves 32-4=28 code points to use for the encoding of ID/AD called ED. The number of possible ID and AD values are higher than the number of available ED code points. If other values than the ones in the table below occurs, a CONTEXT_UPDATE packet MUST be sent instead. The encoding is carried out as follows:

AD	ID	ED
3	1	001 00
4	1	010 00
5	1	011 00
6	1	100 00
7	1	101 00
8	1	110 00
9	1	111 00
3	2	001 01
4	2	010 01
5	2	011 01
6	2	100 01
7	2	101 01
8	2	110 01
9	2	111 01
3	3	001 10
4	3	010 10
5	3	011 10
6	3	100 10
7	3	101 10
8	3	110 10
9	3	111 10
3	4	001 11
4	4	010 11
5	4	011 11
6	4	100 11
7	4	101 11
8	4	110 11
9	4	111 11

This Internet-Draft expires April 22, 2000.