Robust Header Compression (RoHC) WG
Internet Draft
Document: <draft-miyazaki-rohc-kwhc-00.txt>
March 01, 2000
Expires: September 01, 2000

Akihiro Miyazaki
Hideaki Fukushima
Thomas Wiebke
Rolf Hakenberg
Carsten Burmeister

Matsushita

Robust Header Compression using Keyword-Packets

Status of this Memo

1. Abstract

Header Compression has proven to be essential for using RTP over bandwidth limited links. While all of these links have only a limited bandwidth, some are additional extremely unreliable, e.g. wireless links. It is shown, that existing Header Compression schemes, that work well over reliable links, perform very bad, if many errors occur.

This led to the development of a new robust Header Compression mechanism, that is introduced in this paper. The core idea is the definition of keyword packets, which reduces the amount of discarded packets at the decompressor.

The Header Compression scheme, the used packet formats and the decompression algorithm, are described in detail and simulation results of an example application over a wireless network are shown. These results justify the stated assumptions and conclusions.

Table of Contents

2. Conventions used in this document

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 [2].

3. Introduction

The increasing use of real-time services over the Internet, such as voice or video applications, led to a strong demand for a protocol to be used to transport data real-time or nearly real-time. The Real-time Transport Protocol (RTP) provides the means for such services. After the publication as an RFC, more and more applications were developed to use RTP.

Another important development in the last years has been the user’s demand for mobility, i.e. accessing many services, including video and data applications, anytime and anywhere.

While RTP is commonly used in fixed networks, which is the environment it was developed for, using it over mobile networks needs some further considerations. A problem that occurs is the limited bandwidth in the mobile channel. There is a need to use this bandwidth very efficiently to satisfy enough users. Enough users per cell are very important to be economically feasible.

Each of the protocols RTP, UDP and IP adds its own header, which sums up to 40 byte in most of the cases, i.e. 12 bytes RTP header, 8 bytes UDP header and 20 bytes IP version 4 header (If IP version 6 is used, the amount of overhead is increased by another 20 bytes, because of the extended address format).

To solve the problem of the large amount of overhead header compression mechanisms have been developed. CRTP is commonly used over reliable links. It compresses the header to a minimum of 2 bytes. While this works fine over reliable channels, it induces many additional errors, when used over unreliable channels. This is because the first or second order differences to the previous packet, which are efficiently coded, are transmitted. If a packet gets lost, due to bit errors over the wireless channel, the compressed header of the next packet cannot be decompressed. The decompressor must send a request for a packet with a decompressed header to resynchronize, but in the meanwhile all received compressed packets have to be discarded. This effect is shown in detail in [8] and can be seen from the simulation results at the end of this document.

In this paper we would like to present a header compression mechanism that is more robust against packet loss and hence performs better over unreliable channels.

4. The concept of keyword header compression

This section describes the concept of the keyword header compression proposal and how this concept leads to a better performance, when used over unreliable links.

The main idea behind this proposal is the definition of a keyword packet. The packets following a keyword packet contain the difference of their header fields to this reference packet’s header. Hence, if a non-keyword packet gets lost, the receiver is able to decompress the following packet.

While compressing and transmitting the packets, the encoded differences may increase and need more coded bytes in the compressed header. If the header size exceeds a certain threshold, a new keyword packet is defined.

If the keyword packets are transmitted correctly, the receiver is able to decompress any incoming compressed header. However, bit errors may occur while transmitting keyword packets as well. In this case the receiver must send a message to the compressor, to ask for a packet with a header that reestablishes its context. This is the new keyword packet.

5. Packet formats

This section shows the three different packets that are used to transmit the data and signal errors from the receiver to the sender. The format of the packets is described and it is explained in detail how the decompressor is able to regenerate the complete header from the given information. The exact compression behaviour is implementation specific, but it has to be ensured, that any decompressor is able to regenerate the complete header in the described way.

5.1 Full Header packet

The Full Header packet is sent once at the beginning of the session to establish a valid context. It is only sent another time if requested by the receiver. The receiver must request a Full Header packet only if the initial packet was lost, or a severe error occurred, that cannot be solved by a Compressed Packet.

The format of this packet’s header is quite similar to the original IP/UDP/RTP header. However, as described in other header compression papers, such as CRTP [7], the length fields of the IP and UDP packets are redundant. They are usually signalled by the link layer. This enables us to use these fields to signal the header compression specific session context identifier (CID)as follows:

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|C-L|Keyword|        CID        |   First length field
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        CID        |- - - - - -|   Second length field
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

C-L (2 bit): CID Length
  00 - 4 bit
  01 - 12 bit
  10 - 20 bit

The selection of 4 to 20 bit CIDs enables the compressor the setup enough sessions while keeping the overhead to a minimum.

5.2 Compressed packet

The header of the Compressed Packet is divided into a basic header that is transmitted for every packet and header extensions that are only used if necessary.

5.2.1 Basic header

The basic header's format is as follows:

```
  0   1   2   3   4   5   6   7
+...+...+...+...+...+...+...+...+
:      msb of Session CID       :  if 20 bit CID is used
+...+...+...+...+...+...+...+...+
:      msb of Session CID       :  if 20 or 12 bit CID is used
+---+---+---+---+---+---+---+---+
| lsb of S.-CID |    Keyword    |
+---+---+---+---+---+---+---+---+
| M | E |  Sub Sequence Number  |
+---+---+---+---+---+---+---+---+
: E : Extension |               :
+...+...+...+...+               :
:                               :
+...+...+...+...+...+...+...+...+
:                               :
+        UDP Checksum           +  if non-zero in context
:                               :
+---+---+---+---+---+---+---+---+
|          RTP Data             |
:                               :
```

CID:
The first two bytes can be used for the extended session CIDs.

Keyword:
The Keyword field must be present in every packet. The Keyword field has a length of 4 bit, which leads to 16 different keywords. To detect loss of reference packets, which renewed the keyword, it must be incremented by one at each renewal.

M-bit:
The M-bit represents the RTP Marker bit. This is transmitted in every packet, because it is not predictable (the use is different for different RTP payload formats).

E-bit:
This bit indicates if at least one extension is used. The different possible extensions, that are used to transmit information about irregular changes in the header fields, are described in detail in the following sections.

Sub Sequence Number:
The Sub Sequence Number counts the number of non-keyword packets that were sent after the last keyword packet. With this number it is possible to calculate values, such as the RTP Sequence Number, without explicitly transmitting it. How this is done is explained in detail in the following sections.

E-bit (in extensions):
After the basic header, one or more extensions might follow. The presence of the first extension must be indicated by setting the E-bit of the basic header. The presence of the latter extensions is indicated by setting the E-bit of the previous extension.

Extension:
This field shows the type of the present extension. Every extension has its own 3-bit code. The codes are specified in the following section at the description of the respective extension.

UDP Checksum:
If the UDP Checksum is enabled, this field contains the 16-bit Checksum, else it is not present.

5.2.2 New-Keyword extension

This extension is sent every time a new reference packet (and with this a new keyword) is defined. The new keyword is value of the keyword field of the basic header, incremented by one. It is not transmitted in this extension. The decompressor knows, at receiving this extension, that it has to calculate the new keyword, by incrementing it by one. Since a definition of a new reference packet is a possibility to define a new default delta timestamp or default delta IP Identification, these values can be included in this extension as well.

The format of the New-Keyword extension is defined below:

```
  0   1   2   3   4   5   6   7
+---+---+---+---+---+---+---+---+
| E | 0 | 0 | 0 |ddT|TSL|ddI| - |
+---+---+---+---+---+---+-------+
:    default delta Timestamp    :   if ddT = 1
+...+...+...+...+...+...+...+...+
:    default delta Timestamp    :   if TSL = 1
+...+...+...+...+...+...+...+...+
:default delta IP Identification:   if ddI = 1
+...+...+...+...+...+...+...+...+
```

The extension code of this extension is 000 as can be seen from bits 1,2 and 3 of the first byte.

E-bit:
This bit indicates if another extension follows after this one.

ddT-bit:
This bit indicates if a new default delta timestamp is transmitted. If it is set to one, the next bits define the coded format and the default delta Timestamp.

TSL-bit:
This bit indicates the length of the default delta Timestamp field. If it is set to one, two bytes are used, else the first byte only contains the new default delta timestamp. This leads to a range of [0..255] or [0..65535] respectively.

ddI-bit:
This bit indicates the transmission of a new default delta IP Identification value. If it is set to one, a new value is transmitted in the default delta IP Identification field, which enables a range of [0..255].

5.2.3 Delta-IP-Identification extension

This extension is used to transmit an irregular change in the IP Identification field.

Normally, i.e. if no error or reordering occurred on the links before the compression, the IP Identification will change in one of the following two ways:

- increase by a fixed delta value for each packet
- randomly

If it changes randomly, no prediction can be made, and the field should always contain the complete IP Identification value. If it increases by a fixed value for each packet, the new IP Identification value is calculated, from the reference packet's header, the default delta value and the actual Sub Sequence Number:

ID = ID(K) + (ddID * SSN)

ID – IP Identification of the packet to be decompressed,
ID(K) – IP Identification of the last keyword- (reference-) packet,
ddID - default delta IP Identification (the usual difference between
the IP Identification values of two succeeding packets),
SSN - Sub Sequence Number of the packet to be decompressed.

If the sequence of the IP Identifications, as received from the compressor, change irregular (e.g. because a packet was lost on the link before), the difference to the predicted value is transmitted in the delta IP Identification (dID) field. This leads to the following equation to decompress the IP Identification:

ID = ID(K) + ((ddID + dId) * SSN)

If this encoding was not possible (e.g. the result for dID is not an integer), some unpredictable errors occurred on the previous links, and the IP Identification is transmitted as is.

The format of the Delta-IP-Identification extension is given below

```
  0   1   2   3   4   5   6   7
+---+---+---+---+---+---+---+---+
| E | 0 | 0 | 1 |  IDC  |  dID  |
+---+---+---+---+---+---+---+---+
:              dID              :   if IDC = 01
+...+...+...+...+...+...+...+...+
:                               :
+       IP Identification       +   if IDC = 10
:                               :
+...+...+...+...+...+...+...+...+
```

The extension code of this extension is 001 as can be seen from bits 1,2 and 3 of the first byte.

E-bit:
This bit indicates if another extension follows after this one.

IDC – (ID field Coding):
The IDC field signals the type of encoding for the IP Identification.

IDC = 00:
This means that the following two bits are used to encode dID. In many cases single errors, or single reordering of the first order are the reason that this field has to be used. Hence it is proposed to use a coding that gives a set of values such as [-1,1,2,3], because in most of the cases these values apply.

IDC = 01:
This means that the following ten bits are used to encode dID. A coding is proposed that gives a set of values in the range [-511..512]. This is enough to encode most dIDs. If dID is not in this range, the complete IP Identification must be transmitted.

IDC = 10:
This means that the following 2 bytes contain the original IP Identification value. This is necessary, if the dID is not

encodable. The first two bit after the IDC field are not used in this case.

5.2.4 Delta-RTP-Sequence-Number extension

This extension is used to transmit an irregular change in the RTP Sequence Number (SN) field.

The SN normally increases by one for each packet. In this case it is calculated at the decompressor according to the following equation:

SN = SN(K) + SSN

```
SN    -  RTP Sequence Number of the packet to be decompressed
SN(K) –  RTP Sequence Number of the last reference- (keyword-)
packet
SSN   -  Sub Sequence Number of the packet to be decompressed
```

If an error occurred on a previous link, or for some other reason SN was not increased exactly by one, the difference to the predicted SN (dSN) is transmitted, if possible. This leads to the following equation to decompress the RTP sequence Number:

SN = SN(K) + SSN + dSN.

The format of the Delta-RTP-Sequence-Number extension is given below

```
  0   1   2   3   4   5   6   7
+---+---+---+---+---+---+---+---+
| E | 0 | 1 | 0 | L |    dSN    |
+---+---+---+---+---+---+---+---+
:              dSN              :   if L = 1
+...+...+...+...+...+...+...+...+
```

The extension code of this extension is 010 as can be seen from bits 1,2 and 3 of the first byte.

E-bit:
This bit indicates if another extension follows after this one.

L-bit:
This bit signals the format in which dSN is encoded.

L = 0:
This means the following 3 bits are used to encode dSN. For most of the cases a range of [-4,..,-1,1,..,4] is sufficient. A coding is used to achieve the described range.

L = 1:
This means the following 11 bits are used to encode dSN. This leads to a maximum range of [-1023..1024]. In cases where dSN cannot be coded with the previous option, it is tried to apply this option. If it even exceeds this range, something very unusual happened and an update extension (see section 5.2.7), which contains the complete RTP Sequence Number, must be sent.

5.2.5 Delta-RTP-Timestamp extension

This extension is used to transmit an irregular change in the RTP Timestamp (TS) field. The timestamp may increase by a fixed value (default delta timestamp) in most of the cases. However in some cases errors occur and disturb this sequence and in other cases, where a different payload is transported, the timestamp may increase quite randomly and not predictable.

If TS increases by a fixed value, for each packet, we should take advantage of it. At detecting a default delta timestamp, e.g. if TS increased by the same value in n succeeding packets, the compressor could use the default delta timestamp field to signal this value. For the following packets TS could be calculated from the keyword packet’s timestamp (TS(K)), the Sub Sequence Number (SSN) and the signaled default delta Timestamp (ddTS) as follows:

TS = TS(K) + (SSN * ddTS)

In cases where, because of an error or a reordering in the previous link, TS increases different, the difference between the calculated TS is transmitted:

dTS = TS – TS(K) – (SSN * ddTS)

With this information the decompressor is able to regenerate the original header.

The format of the Delta-RTP-Timestamp extension is given below

```
  0   1   2   3   4   5   6   7
+---+---+---+---+---+---+---+---+
| E | 0 | 1 | 1 |   L   |  dTS  |
+---+---+---+---+---+---+---+---+
|              dTS              |
+---+---+---+---+---+---+---+---+
:              dTS              :  if L = 10
+...+...+...+...+...+...+...+...+
:              dTS              :  if L = 11
+...+...+...+...+...+...+...+...+
```

The extension code of this extension is 011 as can be seen from bits 1,2 and 3 of the first byte.

E-bit:
This bit indicates if another extension follows after this one.

L-bit:
This bit signals the format in which dTS is encoded.

L = 00:
This means the following 10 bits are used to encode dSN. For most of the cases a range of [-511..512] is sufficient. A coding is used to achieve the described range.

L = 10:
This means an additional byte is used to code dTS, which leads to a greater range, i.e. [-262143..262144]

L = 11:
This means that two additional bytes are used to code dTS. This leads to a total amount of 26 bit, which is sufficient for nearly all cases, because the original timestamp even consists of only 32 bits. If this coding fails, because the range is exceeded, an update packet, including the original TS is transmitted (see section 5.2.7).

5.2.6 CSRC-Count extension

This extension is used to transmit a change of the RTP-CSRC-Count field. The format of this extension is as follows:

```
  0   1   2   3   4   5   6   7
+---+---+---+---+---+---+---+---+
| E | 1 | 0 | 0 |  CSRC Count   |
+---+---+---+---+---+---+---+---+
:                               :
:           CSRC List           :
:                               :
+...+...+...+...+...+...+...+...+
```

The extension code of this extension is 100 as can be seen from bits 1,2 and 3 of the first byte.

E-bit:
This bit indicates if another extension follows after this one.

The new CSRC Count is transmitted in the last four bits of the first byte. The complete CSRC List follows this extension directly.

5.2.7 Update extension

If a keyword packet was not received at the decompressor (which is detected by a new keyword, without using the New-Keyword or Update extension), the decompressor sends a Context State packet to the compressor (see section 5.3). This packet contains the last correctly received keyword value and the actual received keyword.

Upon reception of a Context State packet, the compressor decides if the context of the decompressor needs an update, or if an update was already sent regarding this keyword. To update the context of the decompressor, a packet with the Update extension is sent.

Another use of this extension could be, if one of the delta values, as described before, is not encodable. This could be, because the value exceeds the maximum range of the extension. In this case, the compressor sends the Update extension, without being requested to do so.

Every Update extension packet is a new keyword packet. This means that the new keyword is calculated by incrementing the transmitted keyword value in the basic header by one.

The format of the Update extension is as follows:

```
  0   1   2   3   4   5   6   7
+---+---+---+---+---+---+---+---+
| E | 1 | 0 | 1 |  CSRC  Count  |
+---+---+---+---+---+---+---+---+
|                               |
+      RTP Sequence Number      +
|                               |
+---+---+---+---+---+---+---+---+
|                               |
+       IP Identification       +
|                               |
+---+---+---+---+---+---+---+---+
|                               |
+                               +
|                               |
+         RTP Timestamp         +
|                               |
+                               +
|                               |
+---+---+---+---+---+---+---+---+
```

The extension code of this extension is 101 as can be seen from bits 1,2 and 3 of the first byte.

E-bit:

This bit indicates if another extension follows after this one.

The remaining fields represent the original fields from the RTP header.

5.2.8 Changing-Fields extension

This extension is used to transmit the change of fields that normally do not change regularly. These are the fields Time-To-Live and Type-of-Service for IP version4 and Hop-Limit and Traffic-Class in IP version6 and RTP-Payload-Type of the RTP header. If changes in those fields occur, it is important to signal them to the decompressor, which is done with the Changing-Fields extension.

It is however important to keep in mind, that the context of the decompressor is only updated to the new values of these fields, if the packet is a keyword packet. Thus if the compressor wants to change the value of the field permanently, a keyword packet has to be sent, including the Changing-Fields extension. Else the values are changed only in this packet, and will not have an effect on the context of the decompressor.

Another thing to keep in mind, is that keyword packets can get lost. If a keyword packet with a Changing-Fields extension was lost, these fields must be retransmitted, to establish the same context at the sender and receiver. The compressor knows from the Context State packet about the keyword(s) of lost packet(s). Hence if a keyword packet with Changing-Fields Extension is lost, it must be retransmitted.

The format of the Changing-Fields extension is as follows:

```
  0   1   2   3   4   5   6   7
+---+---+---+---+---+---+---+---+
| E | 1 | 1 | 0 |TTL|TOS|PT | - |
+---+---+---+---+---+---+---+---+
|   Time-to-Live / Hop-Limit    |
+---+---+---+---+---+---+---+---+
|Type-of-Service / Traffic-Class|
+---+---+---+---+---+---+---+---+
|       RTP Payload Type        |
+---+---+---+---+---+---+---+---+
```

The extension code of this extension is 100 as can be seen from bits 1,2 and 3 of the first byte.

E-bit:
This bit indicates if another extension follows after this one.

The remaining fields represent the original fields from the IP- or RTP- header.

5.3 Context State packet

This header compression proposal is aimed to perform good, even if used over an unreliable channel. Hence bit errors can occur quite frequently and packets will get lost. If the lost packet was a non-keyword packet, this does not effect the decompressor at all, but a lost keyword packet invalidates the decompressor’s context. Hence any compressed packet, even if it was received correctly, cannot be decompressed, until the context is updated correctly again.

To request a context update, the decompressor must send immediately after detecting an invalid context, a Context State packet. This packet contains the last correctly received keyword and the just received keyword, that invalidated the context. The compressor knows at reception of such a Context State packet, what packet it has to send to update the decompressor’s context correctly.

Another error, that could occur, is the loss of the first packet, i.e. the Full Header packet. Since most of the header information, e.g. addresses and ports, are transmitted only once in this packet, it is essential for the decompressor to establish a valid context to receive this packet. If the decompressor receives a Compressed packet, with a new session CID, that was not initialized, by a Full Header packet, this Full Header packet must have been lost. In this case the decompressor must request a new Full Header packet, by the means of the Context State packet.

The format of the Context State packet is as follows:

```
+...+...+...+...+...+...+...+...+
:      msb of Session CID       :  if 20 bit CID is used
+...+...+...+...+...+...+...+...+
:      msb of Session CID       :  if 20 or 12 bit CID is used
+---+---+---+---+---+---+---+---+
| lsb of S.-CID |RFH|           |
+---+---+---+---+---+---+---+---+
:   Keyword 1   :   Keyword 2   :  if RFH = 0
+...+...+...+...+...+...+...+...+
```

CID:
The first two bytes can be used for an enhanced CID.

RFH (Request Full Header):
If this bit is set to one, the first Full Header packet was lost, no context was established and a new Full Header packet is requested.

If it is set to zero a context update is required and the two keyword fields are present.

Keyword 1:
This field contains the last correctly received keyword.

Keyword 2:
This field contains the most recent received keyword, that invalidated the context.

6. Performance evaluation of the proposed solution

The previous sections introduced a new header compression mechanism and it was stated, that it would perform better over unreliable links. This was justified by the newly introduced keyword mechanism, which makes the packets independent from their predecessor. Many packets depend on one reference packet, which makes the mechanism more robust against packet loss.

This section shows simulation results to prove the above statements. As an example application, we choose to simulate a voice over IP application in a wireless communication network. This was done, to be, at least qualitative, comparable to [8].

6.1 Application

In this performance evaluation we simulate a voice over IP application. A GSM enhanced full-rate codec is simulated that produces 50 packets per second (each packet has a fixed payload size of 256 bits). The used traffic model generates talk spurts and silent periods with lengths, which are both exponentially distributed with an expected length of one second. During silent periods, no data is transmitted (Silence Suppression).

6.2 Environment

A mobile environment is considered as a typical representation for unreliable, bandwidth limited links. The used protocols for the evaluation are shown in Figure 2.

```
  Media Server           Mobile Network         Mobile Client

  +-----------+                                  +------------+
  |Voice o. IP|                                  |Voice o. IP |
  |Application|                                  |Application |
  +-----------+                                  +------------+
  |    RTP    |                                  |    RTP     |
  +-----------+                                  +------------+
  |    UDP    |                                  |    UDP     |
  +-----------+        +------------+            +------------+
  |    IP     |<--------->|     IP     |            |     IP     |
  +-----------+        +------------+            +------------+
                       |Header Comp.|            |Header Comp.|
                       | Compressor |            |Decompressor|
                       +------------+            +------------+
                       |   W-CDMA   |<--------->|   W-CDMA   |
                       +------------+            +------------+
               Internet                  Wireless
               Link                      Link
                          Figure 2
```

The IP network is modeled with a uniformly distributed error probability of 0.01 and a uniformly distributed reorder probability of the first order of 0.01.

For the wireless link a W-CDMA channel is considered. W-CDMA, as a third generation mobile communication system with high Quality-of-Service (QoS) capabilities, is a well suited network for this kind of applications in a mobile use.

The W-CDMA system is standardized at the 3rd Generation Partnership Project (3GPP). The standardization is still under progress and no final specifications are finished. For the simulations the specifications of December 1999 are considered. The wireless link and its link layer protocols are simulated according to this specifications (see [9] for details).

The channel is simulated with the fading model, as described in [10]. The channel has a mean signal-to-noise ratio per received symbol of Es/N0. Different mean Es/N0 values are simulated, which reflect different channel conditions.

The return channel in our simulations is modeled as error-free, loss-less.

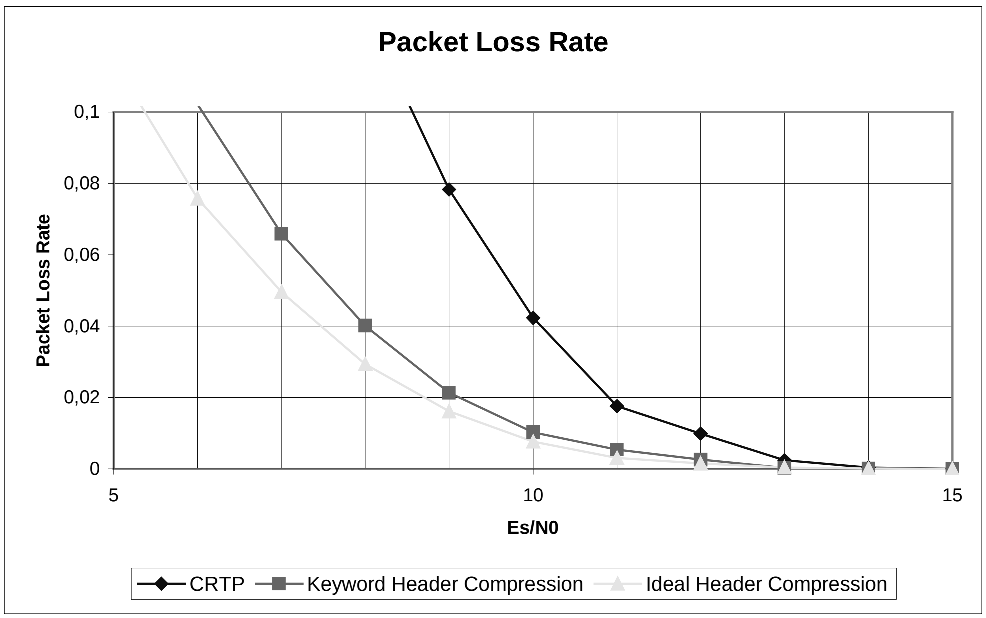
The propagation delay of the mobile channel is dependent on the length of the packet that is to be sent. The minimum delay of 50ms applies to all transmitted frames, due to the coding, interleaving etc. in the physical layer. Since the packets are quite small (i.e.

they will fit in one or two (size limited) link layer PDUs), the additional delay is quite small as well, and is in the range of ms.

6.3 Performance results

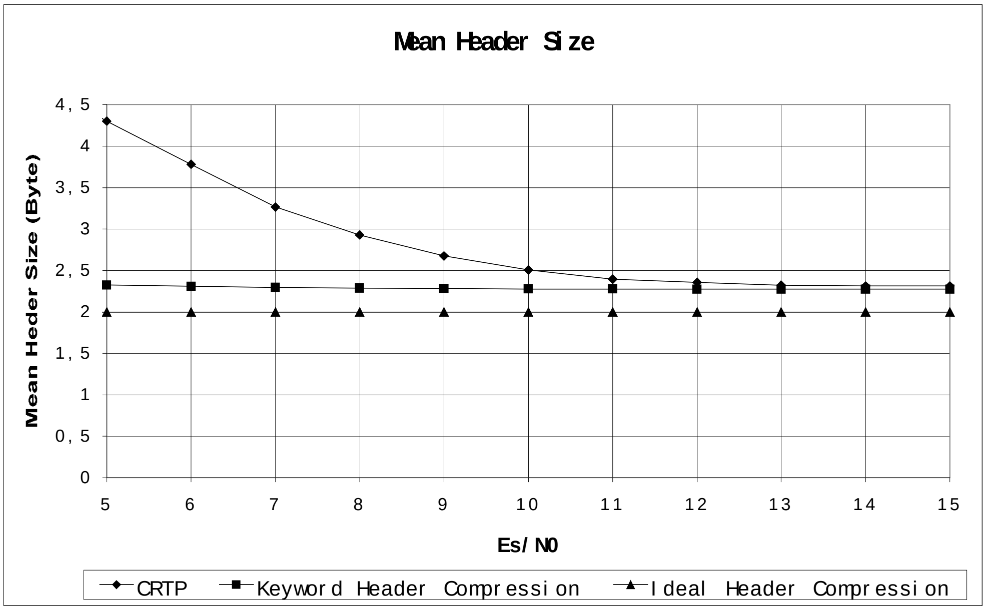
This section shows the results of the simulations as described above. Three different header compression mechanisms are compared. That is: CRTP as described in [7], the keyword header compression as described in this document and an ideal header compression mechanism, which is just imaginary and for comparison purposes. The ideal header compression mechanism compresses every header to two bytes and does not induce any additional error.

The most important result, we wanted to achieve, was reducing the amount of packets that are discarded at the decompressor. The following figure shows the simulated packet loss rate, i.e. the number of packets that are not received or could not be decompressed correctly divided by the total number of packets sent over the mobile channel.



As we can see clearly from this figure, CRTP induces many additional errors (i.e. compared to the unavoidable errors for ideal header compression) in bad channel conditions. These additional errors are reduced by a huge amount for keyword header compression, which justifies our assumption. The definition of a keyword packet reduces the amount of additional errors.

Another result, that should be received, was a good compression efficiency. We define the compression efficiency as the mean header size. The following figure shows the mean header sizes in different channel conditions.



For ideal header compression, the mean header size is two bytes, regardless of the channel condition.

CRTP compresses the header very efficiently as well, if the channel is in a good state, i.e. only very few bit errors occur. But for bad channel conditions, the compression efficiency of CRTP decreases. It can be explained, by the frequently requested Full Header packets, that are needed to update the context. One of these packets is sent for nearly each lost compressed packet.

This effect can be reduced to nearly zero, using keyword header compression. If a non-keyword packet gets lost, the header size of the next packet is not affected. Only if a keyword packet is lost, the context of the receiver is invalidated, and an Update packet is sent, which has a bigger header than the compressed packet. However it is much smaller than a Full Header packet.

Hence, the compression efficiency of our proposal is nearly as good as the ideal header compression’s efficiency and performs much better than CRTP in bad channel conditions.

6.4 Conclusions

It was stated that CRTP induces too many additional errors, when used over unreliable links. A new solution, keyword header compression, was described, which should reduce the number of additional errors.

This could be proven by the simulation results, which show that with the keyword header compression mechanism we move much closer in the direction of ideal header compression.

The compression efficiency, i.e. the mean header size that is achieved, is another important measurement to judge the quality of the header compression mechanism. The simulation results show that the mean header size increases with bad channel conditions up to four byte and bigger, when using CRTP. The keyword header compression's mean header size is nearly not effected by the channel condition and is always smaller than for CRTP.

Hence the simulations prove the stated assumptions and conclusions and show that keyword header compression is suitable for the use over unreliable links.

7. Security Considerations

Security issues are not considered in this memo.

8. Intellectual property considerations

Matsushita has filed patent applications that might possibly have technical relation to this contribution.

9. References

1 Bradner, S., "The Internet Standards Process -- Revision 3", BCP 9, RFC 2026, October 1996.

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

3 Schulzrinne, H., Casner, S., Frederick, R. and V. Jacobson, "RTP: A Transport Protocol for real-time applications", RFC 1889, January 1996.

4 Postel, J.,"User Datagram Protocol", STD 6, RFC 768, August 1980.

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

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

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

8 Mikael Degermark, Hans Hannu, Lars-Erik Jonsson, Krister Svanbro, "CRTP over cellular radio links", Internet Draft (work in progress), June 1999.

9 Homepage of 3GPP: http://www.3gpp.org

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

12. Author's Addresses

Akihiro Miyazaki
Matsushita Electric Industrial Co., Ltd
1006, Kadoma, Kadoma City, Osaka, Japan
Tel. +81-6-6900-9192
Fax. +81-6-6900-9193
Mail akihiro@isl.mei.co.jp

Hideaki Fukushima
Matsushita Electric Industrial Co., Ltd
1006, Kadoma, Kadoma City, Osaka, Japan
Tel. +81-6-6900-9192
Fax. +81-6-6900-9193
Mail fukusima@isl.mei.co.jp

Thomas Wiebke
Panasonic European Laboratories GmbH
Monzastr. 4c, 63225 Langen, Germany
Tel. +49-(0)6103-766-161
Fax. +49-(0)6103-766-166
Mail wiebke@panasonic.de

Rolf Hakenberg
Panasonic European Laboratories GmbH
Monzastr. 4c, 63225 Langen, Germany
Tel. +49-(0)6103-766-162
Fax. +49-(0)6103-766-166
Mail hakenberg@panasonic.de

Carsten Burmeister
Panasonic European Laboratories GmbH
Monzastr. 4c, 63225 Langen, Germany
Tel. +49-(0)6103-766-263
Fax. +49-(0)6103-766-166
Mail burmeister@panasonic.de

Full Copyright Statement