

Multiplex header compression for transparent cross-layer design

Catherine Lamy-Bergot, Member, IEEE, and Pierre Vila

THALES Communications France *

TRS/TSI

160 boulevard de Valmy F-92704 Colombes Cedex

catherine.lamy@fr.thalesgroup.com

phone: +33 1 46 13 27 90

fax: +33 1 46 13 25 55

Abstract — *The transmission of video or images over bandwidth-limited channels faces the conflicting requirements of being affected by error-prone channels while having at disposal only limited bandwidth resources. Due to these strong constraints, bit-rate reduction strategies such as source coding or header compression have been developed and new solutions relying on an end-to-end optimisation are beginning to appear. A major difficulty comes from the fact that applications do not deal directly with the access level but through protocol networks, which leads to the necessity of transparent cross-layer design. A multiplex mechanism is proposed that takes advantage of the presence of an header compression mechanism such as ROHC (RObust Header Compression) to allow optimisation of application level source coding and FEC at the network access layer in a standard protocol stack, independently of the internet and transport layers.*

Index Terms — *cross-layer design, wireless channel, header compression, joint source channel (de)coding.*

I. INTRODUCTION

THE transmission of text, images and video over bandwidth-limited channels can be dramatically affected by the errors caused by the channel. To combat these errors, applications classically use on one hand source encoding to reduce the redundancy inherent in the source symbols, *e.g.* compression standards relying on variable length codes (VLC: Huffman codes, arithmetic codes,...), and on the other hand channel encoding and modulation, *i.e.* forward error correction (FEC) to increase the robustness of the transmission over the channel. A more integrated optimisation can be achieved by developing a joint source channel en/decoding strategy, which will allow to use appropriately the residual source redundancy at the decoding part to provide error correction capability [1].

In simple systems where the source and channel encoders (*resp.* decoders) are directly interfaced, the different joint source channel coding/decoding techniques can be easily implemented by exchanging information between the different

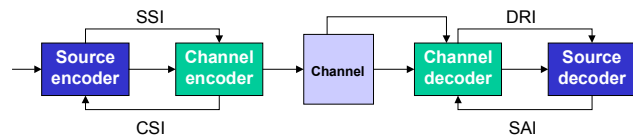


Fig. 1. Generic joint source channel coding scheme.

blocks, as shown in Fig. 1.

At the transmitter side, Source Significance Information (SSI), giving information about the sensitiveness of the source to channel errors, may be passed from the source encoder to the channel encoder to enable techniques like Unequal Error Protection (UEP). In order to adapt source and channel coding rates to channel conditions, it is also useful to provide information about the Channel State Information (CSI) back to both the source and channel encoder. In the digital communication world, the traditional decoding methods applied to such concatenated schemes achieving high coding gains with reasonable complexity and robustness to transmission errors can be either hard-decision or soft-decision, depending on whether the input signal is binary or analog. As soft-decision decoding methods achieve an asymptotic error performance improvement of several decibels on most channels [2]-Ch.8, providing soft information appears almost necessary in modern communications techniques. The inner decoder must then provide soft output to the outer decoder and vice-versa in the case of an iterative decoding. As a consequence, the channel soft outputs and CSI, related to both the fading amplitude and the noise, may be provided at the receiver side by the channel to the channel decoder to perform soft-input soft-output channel decoding. The obtained channel decoder soft output, or Decoder Reliability Information (DRI) will then be given to the source decoder which will perform soft-input source decoding and eventually send its soft output, or Source A posteriori Information (SAI) back to the channel decoder.

However, actual source and channel en/decoders are often networked devices that can not exchange information due to the protocol layers that separate them, as shown in Fig. 2.

The various information to be exchanged between the decoders must consequently be passed through different levels of network protocols. Then, to remain compliant with existing

* This work was supported in part by the French government through project RNRT-VIP (convention 02-2-93-0241).

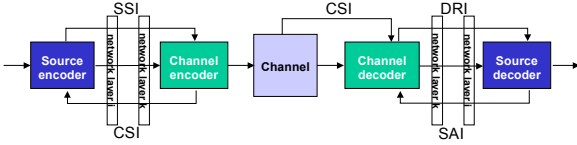


Fig. 2. Generic joint source channel coding scheme.

networks recommendations and implementations [3][4], such transmissions should not interfere with the regular use of the network. This implies that the extra information should be transmitted transparently for the protocol stack, which will allow for the system to be backward compatible.

To exchange information between layers without modifying the protocol stack, a first idea would be to bypass said stack and use a parallel link, in particular when working on the same machine. As a matter of fact, some links do exist in practice on a computer between the kernel (lower layers) and the user-level (higher layers), which allow a few specific exchanges between the kernel and the user-level without going through the protocol stack. Dedicated drivers with *ioctl* links [5] allow the application layer to capture and filter data at data link level. Such a solution is however only applicable locally, and corresponds to a case where the data does in fact not go through the protocol stack.

Another solution was proposed to allow for the exchange of information such as the reliability (or soft information) through a network between the channel and source decoders, with avoiding any interference with the standard network use, and consequently any redefinition of the existing protocols. Presented in [6], this *network transparency* technique relies on the presence of two adaptation layers, which allow to take into account the existing QoS mechanisms, and on the validation of the concept in a Berkeley Software Distribution Linux environment. This solution has the advantage to be adaptable to any protocol stack and any network, but imposes first to have perfect knowledge of the protocol stack both at the network access and application level, and second remains rather complex to implement.

Following the path opened by this previous solution, the present article proposes to take advantage of the presence of an header compression mechanism in standardised wireless communication such as UMTS and use this mechanism to build or modify the content of valid IP packets at the network access level. A solution is described that allows the exchange of information (CSI, SSI, DRI, SAI, ...) between the source and channel decoders in presence of intermediate network layers without any modification of said layers. This information exchange will permit to improve the decoding performance in the context of real-time data transmission with header compression over a mobile network, for instance thanks to the transmission of bits reliability. This solution presents the double advantage to be backward compatible with existing IP based networks that implement header compression, hence allowing to use QoS management tools proposed by the IETF and to embed the transmission of extra information within the

header compression construction and reconstruction mechanism for a reduced increase of numerical complexity.

This article is organised as follows. Section II presents the networked transmission scheme, describing the implications of the OSI layer model and the header compression mechanism. Section III introduces the proposed technique to realise the needed information exchanges via the generation of extra valid packets at the header compression level. Practical example of the generated header fields and modification for original packets in the RTP/UDP/IPv4 context are proposed. Finally, Section IV draws out the conclusions.

II. NETWORKED TRANSMISSION SCHEME WITH HEADER COMPRESSION

For image and video transmissions using Internet protocols, many solutions have been proposed and are currently deployed or under study. Typical solutions are based on protocols TCP or UDP, with the real-time transmission dedicated Real-time Transfer Protocol (RTP). The stack considered in the following to illustrate the proposed technique will consequently be an RTP/UDP/IPv4 one.

A. Influence of the OSI layer model: revisiting the basics

In practice, the transfer of extra information between the channel and source decoders will consist in the transmission of their quantified values through the network protocol stack. The problem becomes then to transmit several binary inputs (typically 4 or 5) instead of one for each information bit considered. However, as they are not transmitted directly but through a network, the information bits that interest the application constitutes only a part, or payload of the effectively emitted sequence. As illustrated in Fig. 3, this



Fig. 3. Syntax principle for a networked sequence transmitted over the physical channel.

sequence is composed by the payload encapsulated by headers and eventual trailers (*e.g.* check fields).

More explicitly, the transmission of the data downward through the protocol stack will consist at each layer border in the following steps [3]:

- getting the data sequence S_{L+1} from upper layer,
- generating the *ad hoc* header and eventual trailer,
- creating the new data sequence S_L by concatenating the header, the sequence S_{L+1} and the trailer. This step eventually leads to cutting the data sequence S_{L+1} into several blocks due to size limitations imposed by the protocols. In that latter case, the resulting packets keep a similar constitution to the not-divided ones, and can hence be treated similarly in the proposed scheme.

On the other side of the channel, the upward transmission through the protocol stack will consist at each layer border in:

- getting the data sequence S'_{L-1} from lower layer,
- decapsulating the *ad hoc* header (and eventual trailer)

to create the sequence S'_L . This step is eventually performed together with retransmission requests when the decapsulation indicates that the data flow was corrupted,

- forward the data sequence S'_L to upper layer if the check field is correct.

B. Header Compression: goals and implementation

The wireless link is characterised by a limited bandwidth, which in practice limits the information bit-rate to be received/transmitted by the user. As a matter of fact, the wireless link is traditionally seen as a bottleneck for the transmission, especially as the high bit and frame error rates (BER and FER between 10^{-2} and 10^{-5}) often lead to multiple retransmissions which worsens the bandwidth scarcity problem. As a consequence, the direct transmission of IP packets over the wireless physical links leads to a dramatic waste of the information bit-rate. As a matter of fact, the headers of RTP, UDP and IP layers add a non-negligible load to the information payload, load that can easily be reduced as these headers are often redundant either within the header itself or with the previous and following ones.

To answer this double goal of header reduction and header robustness increase for wireless links, a new protocol was proposed by the IETF, that has been introduced in the UMTS releases 5 and 6 by the 3GPP working group. This scheme, named RObusT Header Compression (ROHC) has been standardised by the IETF under RFC 3095 [7]. Its principle is to compress the transport and network headers by transmitting only the non-redundant information. Standardisation studied for RTP/UDP/IP header compression in UMTS link are currently being carried out by the IETF [8][9].

To better illustrate this ROHC mechanism, let consider the case depicted in Fig. 4. The various header fields in the IPv4, UDP, RTP protocol stacks can be classified as follows:

- INFERRED: data that can be directly derived from the other headers fields. They are not transmitted.
- STATIC: fields static through the whole data transmission. They are sent only once.
- STATIC-DEF: fields defining the data flow. They are managed like STATIC ones.

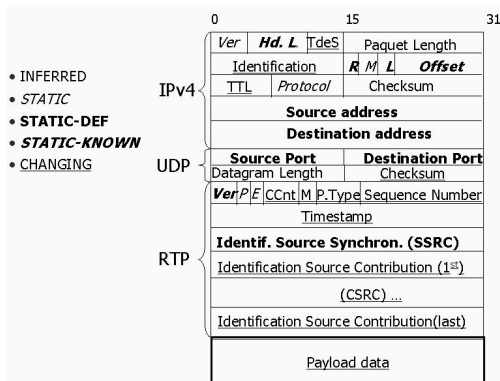


Fig. 4. Various header fields for a RTP/UDP/IPv4 stack and their classification.

- STATIC-KNOWN: fields with known values. They are not transmitted.
- CHANGING: fields whose values change regularly, whether randomly or periodically. They are fully transmitted.

III. TRANSMITTING INFORMATION VIA EXTRA VALID PACKETS

The idea presented in this paper is that the extra information to be transmitted from the network access level to the application level should be packetised by the header compression reconstruction module at the same time as the original transmitted data. This integration within the reconstruction process allows for the hypothesis that the syntax to be used to build extra packets is known, and that the syntax of the reconstructed original data packets can be modified with respect to the user will. Similarly, the extra information to be transmitted from the application level to the network access level is transmitted via extra packets that are intercepted by the header compression module and extracted to be used with respect to the user needs. Those extra packets are then multiplexed with the original data flow for transmission through the standard protocol stack.

Working in an error-prone environment where the bandwidth is limited, *i.e.* where any extra information can allow to improve greatly the transmission, the proposed technique allows to:

- locate the information needed to generate extra valid packets headers and eventually modify the original packets headers according to the user needs at the network access level;
- extract the extra information present at the network access layer and use it as payload data for the extra valid packets sent over a dedicated port to the application level;
- generate extra valid packets at the application level to provide extra information at the network access layer;
- extract the extra information sent by the application level at the network access level.

A. Generating extra valid packets headers to carry extra information to the application level

As highlighted when presenting the interest of header compression with the example of ROHC mechanism [7], header compression relies on the knowledge that the various header fields in the RTP/UDP/IP protocol stacks have a fixed syntax that can easily be reconstructed from only partial information (typically STATIC, STATIC-DEF and CHANGING classes).

Having observed that fact, it is easy to understand that the reconstruction mechanism can also in parallel to the data flow build extra packets based on the same headers fields. Fig. 5. shows an example of the application of this principle, with three classes of headers fields for extra packets:

- RECOPIED: they correspond to fields that are directly copied from valid packets data. In practice, these fields belong principally to STATIC, STATIC-DEF and

STATIC-KNOWN data, but can also be CHANGING one recopied as such (e.g. Timestamp).

- **INFERRED**: as in standard ROHC process, these fields are directly derived from the other headers fields.
- **SPECIFICALLY DERIVED**: these fields are those that are specifically modified in order to allow for the extra information transmission process. In particular, we find:
 - the Destination Port, that will allow the user to separate the original data flow from the extra one, and avoid disturb the standard functioning of the various protocols (e.g. RTCP). It is proposed for instance to transport the original data and the extra information over two distinct transport port numbers;
 - the UDP Checksum, which depends of the Payload data, hence must be re-derived for the new payload these extra packets will carry;
 - the Sequence Number, which will be used to identify the original packet this extra packet corresponds to;
 - the Payload data, which will be replaced by the extra information we want to transmit.

In practice, the values for these specifically modified fields

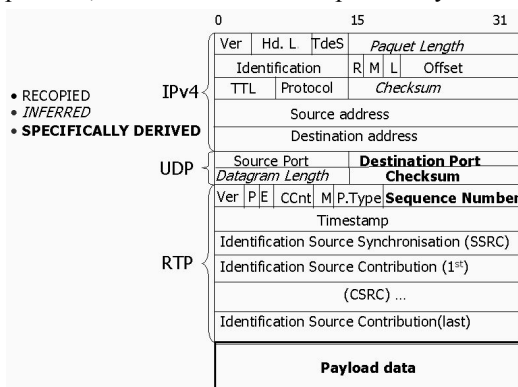


Fig. 5. Generation of header fields for extra packets in a RTP/UDP/IPv4 stack and their classification.

can be set by the user depending on his requirements. An example of possible rule to fill these fields is as follows:

- the Destination Port can be chosen whether dynamically, for instance as the first one directly available above the original data transmission one, or be registered as dedicated port for such transmission, for instance by the IANA organisation (<http://www.iana.org>);
- the Sequence Number can be used to identify the original packet this extra packet corresponds to in case of a reliability information or a set of packets to which it can be applied in case of CSI, SSI... In the last case, the sequence number of the first packet to which it can be applied may be used. In the first case, a simple formula such as $SeqNum_{ep} = k * SeqNum_{od}$; can be used, where k is the number of quantification bits, and $SeqNum_{od}$ is the sequence number of the original data packet;

- the payload data can be derived by quantifying the extra information, as detailed in Section III-C. In the case of reliability information (DRI or SAI), a typical value for k is 4 or 5, the first quantification bit being for instance taken equal to the hard value (original data estimation) for better efficiency. For extra information such as SSI or CSI, the format should be specifically pre-determined between the two coders, for instance with two information bits for the CSI, coded over four levels: very bad, bad, good and very good.

B. Modifying original packets accordingly to the user needs

Beside the extra packets, it may be useful for the user to modify the original packets headers. As a matter of fact, it is easy to see that the reconstruction process of the headers can easily be adapted to specific needs of the transmission with extra information. For example, the checksum fields over the payload (e.g. the UDP checksum) can be disabled by being set to zero. In that case, an error in the payload part of the packet won't lead to the discarding of this packet whose payload may be corrected thanks to soft source decoding.

An example of such a modification is given in Fig. 6.

Note that such modifications won't disturb the information transmission, as the original packet is transmitted normally through the protocol stack. If there are no error in any protocol headers, the packet goes through the whole stack and is received at the application level, which ensures that, if used, RTCP packets are sent normally and guarantee the QoS of the transmission.

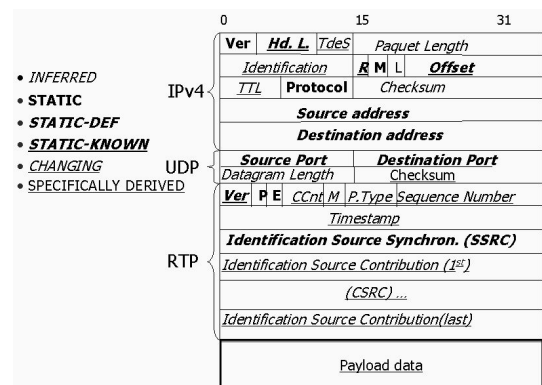


Fig. 6. Example of header fields modification for original packets in a RTP/UDP/IPv4 stack.

C. Extracting extra information present at the network access layer and integrating it in extra valid packets for further use at the application level

Several cases can be envisaged for the extra information to be transmitted: whether CSI, DRI or any other, this information must be formatted to fit within the extra packets. As any other, these packets will transport bits, which implies that the extra information must be consequently translated into bits.

In the case of decoder reliability information, or any information directly proportional to the original information payload, this leads to the necessity of a quantification [2]-Ch.3 process over a given number k of bits. In the case of channel state information, or any information of size not proportional to the information payload (and in practice quite short), this implies a quantification or modelling process over a given number k of bits.

This information extraction made, the quantified values are to be transmitted in parallel to the standard flow to the header compression module, which will exploit them.

D. Generating extra packets at the application level for the network access level and extracting it from the extra packets at the network access layer

Let now consider the transmission of extra information from the application level to the network access one. In particular, as shown in Fig. 2, SSI or SAI can be exploited at the network access level. For systems using header compression, this is feasible by generating extra packets at the application level, that will contain the extra information. Those packets can then be sent via a dedicated port number, similarly to the network access to application case. Of course, those packets will be sent without any ARQ capability, as they won't be actually transmitted but intercepted at the network access layer. At network access level, the header compression module that traditionally performs the header compression, and has in consequence knowledge of the packets structure is modified to test the presence of the dedicated port:

- if the dedicated port is found, the module recognises the extra packet as such and removes it from the data flow. The payload is then extracted to be used by the channel decoder (demodulator, ...).
- if the port is not part of the dedicated ones, the standard mechanism is applied.

The exchanges that take place with the proposed technique are summarised in Fig. 7 for the transmitter side, and in Fig. 8 for the receiver side. The symmetry of these exchanges and the embedding of the performed treatments within the header compression modules are highlighted. This strong integration within the standard process shows that the proposed technique can be easily integrated in any system using header

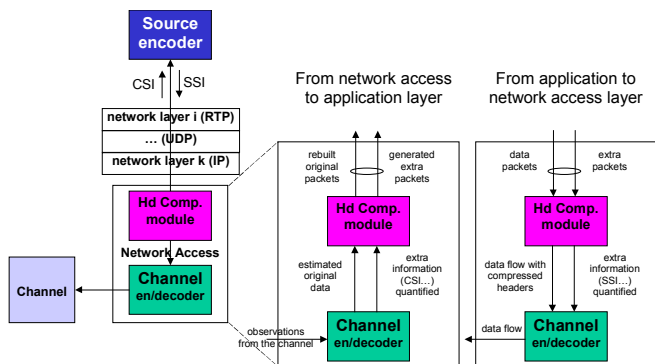


Fig. 7. Exchanges at the transmitter side with extra information transmission capacity.

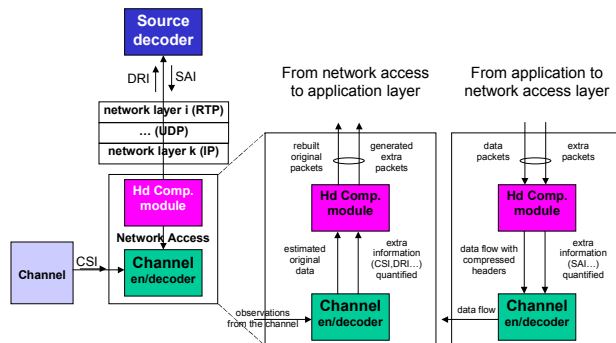


Fig. 8. Exchanges at the receiver side with extra information transmission capacity.

compression at the price of a few modification of the channel coder, source coder and header compression module and are fully compliant with existing IP networks. Moreover, the treatments done to implement the proposed technique do not introduce an important delay, in the sense where after the header reconstruction for the first original packet, the generation of the estimated original packet and of all the extra packets (up to k) can be done in parallel.

IV. CONCLUSIONS

This article presents a technique for exchanging extra information between the application and the network access levels in a transparent way through a network protocol stack for image or video transmissions. This solution is to be considered for low bit-rate networks such as wireless ones built over a network protocol stack with header compression capacity like an RTP/UDP/IP stack implementing ROHC. It consists in generating extra valid packets containing the supplementary information one wants to transmit and multiplex those packets with the original data stream.

Naturally, the interest of this solution comes from the high error rates over wireless links, that can be reduced thanks to the exploitation of any new information. This technique can also be interesting in networks with very long Return Time Trip (RTT), where the use of CSI could help the behaviour of the source decoder to choose between requesting retransmission or applying concealment techniques or other treatments to the received data, depending on the chance it has to receive a correct information at its second request.

This technique can also be beneficial when information about the source bitstream flow such as the SSI can be provided by source encoder to the channel encoder, as it is the case when Unequal Error Protection (UEP) is performed at the network access level.

REFERENCES

- [1] L. Perros-Meilhac and C. Lamy, "Huffman tree based metric derivation for a low-complexity sequential soft VLC decoding," *Proceedings of ICC'02*, vol. 2, pp. 783-787, New York, USA, April-May 2002.
- [2] J.G. Proakis. *Digital Communications*. McGraw-Hill Book Company, New York, 3rd edition, 1995.

- [3] A. Tanenbaum. *Computer networks*. Prentice-Hall, New-York, 3rd edition, 1996.
- [4] W.R. Stevens. *TCP/IP Illustrated volume 1: the protocols*. Addison Wesley Professional Computing Series, Jan. 1999.
- [5] A. Rubini. *LINUX Device Drivers*. O'Reilly and Associates, 1998.
- [6] S. Mérigeault and C. Lamy, "Concepts for Exchanging Extra Information Between Protocol Layers Transparently for the Standard Protocol Stack," *10th International Conference on Telecommunications (ICT'2003)*, February 23 - March 1, 2003, Tahiti, French Polynesia.
- [7] C. Bormann *et al.*, "RFC 3095: RObust Header Compression (ROHC): framework and four profiles: RTP, UDP, EPS, and uncompressed", July 2001
- [8] M. Degermark, "RFC 3096: Requirements for robust IP/UDP/RTP header compression", July 2001.
- [9] K. Svanbro "RFC:3409: Lower layer guidelines for robust RTP/UDP/IP header compression," Dec. 2002.