A New Header Compression Scheme for TCP Streams in IP Based Wireless Networks‡

Gennaro Boggia, Pietro Camarda*,‡, Sandro Petrizzelli and Vito G. Squeo

Politecnico di Bari - DEE, V. Orabona 4, 70125 Bari, Italy

SUMMARY

The current trend in the telecommunication world is the proposal of a universal telecommunication system deriving from the convergence between mobile telephony and data transmission. To achieve such a convergence in an efficient way, it is presumably necessary to utilize a network platform totally based on TCP/IP architecture (an “All-IP Network”). However, the encapsulation process of TCP/IP wastes a substantial part of the radio channel bandwidth for the transmission of control information (header) that does not have any specific function at radio level. Since this bandwidth is generally the most expensive and limited resource for wireless networks, it is necessary to adopt a header compression scheme that can reduce the protocol overhead while working efficiently and robustly under noisy links with time-variant BER conditions.

In this paper, a new header compression scheme for TCP streams as a specific header compression profile within the IETF ROHC (RObust Header Compression) framework is proposed and analyzed through simulations exploiting Network Simulator (ns-2). The proposed scheme, compliant with IETF requirements for TCP/IP header compression, is based on the distinct management of DATA and ACK streams associated with the TCP connection and exploits a robust header encoding technique for reducing the effect of error propagation.

The results obtained, in terms of throughput, overhead and goodput, are reported as a function of bandwidth and BER. A comparison with the results obtained without compression allows us to evaluate the performance of the proposed compression scheme in a realistic and dynamic environment.

KEY WORDS: Wireless Systems; Header Compression; ROHC; TCP; IP; 3G.

1. Introduction

The convergence of information technology, telecommunications, Internet and multimedia applications experienced during the last decade have completely transformed the

*Correspondence to: Pietro Camarda, Politecnico di Bari - DEE, V. Orabona 4, 70125, Bari, Italy.
†E-mail: camarda@poliba.it
‡Preliminary aspects of this paper were presented at ICC’02 and MILCOM’02.

Contract/grant sponsor: This work was funded in part by TILAB (Telecom Italia LAB). Any opinions, findings or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of TILAB.
telecommunication world. During this period, telecommunication operators have focused on establishing new networks designed to promote universality in the transport of voice, data and video services, providing a wide variety of services independent of user location. Wireless systems of the Third Generation (IMT-2000) will be the most important implementation of these concepts [1]: a true mobile service convergence of voice, data and images.

Since these systems will be deployed in expensive licensed frequency bands, they are being engineered to provide optimal use of wireless links that notoriously suffer strong limitations in terms of bandwidth, delay and BER and are, at the same time, the most expensive resource for the providers. Thus, a fundamental aspect for the design of such systems concerns the choice of the network platform: to obtain an efficient integration of all foreseen and future services with a Quality of Service (QoS) that can be comparable with the one offered by fixed networks. It is expected that such integrated systems will exploit a network platform totally based on an IP architecture (an All − IP Network) [2] [3] [4]. Thus, a complete and efficient support for all TCP/IP protocols (TCP, UDP, RTP, and so on) is needed not only over the wired network infrastructure (today often based on IP), but also over the wireless links that connect this infrastructure with mobile units (in other words, mobile units and base stations have to efficiently transmit and receive IP packets over radio links).

However, running TCP/IP protocols over a wireless channel can be difficult. The most important reason regards the large header overhead introduced by these protocols in the process of data packetization/segmentation. In fact, as is well known, each layer of the hierarchical TCP/IP architecture introduces control information (header), into each packet, and thus the bandwidth required by any application is always distributed between user data (payload) and control information. For example, a TCP/IP header is at least 40 bytes long for IPv4 and 60 bytes for IPv6 [5], while, with mobile IPv6 [6], the header grows to 100 bytes: the smaller the size of the payload data, the more it is impacted on by the header size. Some applications are based on a protocol suite that produces a payload size that can be comparable or even smaller than header size: for example, in a typical Voice over IP application, the 66-75% of bandwidth required by the service is burned for header transmission and only the remaining 25-33% of bandwidth is used for the transmission of voice information to the final user. Thus, for applications characterized by a small payload, a large header introduces, in a wireless environment, a significant extra delay that can make the service on the TCP/IP platform completely unrealizable.

Moreover, some protocols of TCP/IP stack show other specific problems in wireless environments. For example, as regards TCP protocol in such environments, it has been shown that the limited performance (in terms of low bandwidth, high delay and BER) of wireless channels can result in high data throughput reduction and very high data delivery delay [7] [8]. This degradation is due not only to the large overhead of IP streams, but also to the congestion control mechanisms of TCP, that have been designed for low-error rate communication media and thus do not react very well to the relatively high packet loss rate over wireless channels. However, this effect is reduced running TCP over error-controlled links [9] [10].

To avoid wasting radio channel bandwidth transmitting control information that does not have any function at radio level, header compression can be introduced. In fact, through the reduction of the header size, it is possible to have a lower bandwidth demand (lower cost) for the same application as well as greater efficiency and responsiveness.

Header compression is feasible due to the presence of a significant redundancy among header fields, both within the same packet header but, in particular, among consecutive packets.
belonging to the same stream: by only transmitting static field information a limited number of times during stream duration and by utilizing dependencies and predictability for other fields, the header size can be significantly reduced for most packets.

Several approaches have been proposed to perform this task. In the early nineties, Van Jacobson (VJ) proposed one of the earliest significant examples of header compression [11] able to manage the header compression of a specific TCP/IPv4 stream on a low-speed wired link. Compressing the 40 bytes header to 3-5 bytes in the common case, VJ TCP header compression is efficient and surely the most widely deployed header compression protocol. However, VJ compression works only with TCP streams and derives its strength from rigid specifications. Moreover, its performance on a high BER wireless link degrades considerably [12] [13].

A number of other protocols have since been proposed. Of particular note is the work of Degermark et al. [12], the first example of a single scheme of header compression able to manage a wide variety of streams defined in a TCP/IP architecture. Particular attention is paid to IPv6 and to real time streams (based on a RTP/UDP/IPv4 and RTP/UDP/IPv6 protocol stacks).

Other detailed specifications for the header compression of RTP [14] [15], the basic protocol for real time streams, are described in [13] and [16]. However, each of these proposals specifies a solution for a given protocol: as a result, they cannot constitute a valid solution for the implementation of a unique header compression scheme able to manage all possible streams (also future) defined within the TCP/IP platform.

Moreover, performance of these header compression schemes are quite poor on wireless links, essentially due to the high BER of a wireless channel (e.g., $10^{-2} \div 10^{-3}$) and high Round Trip Time (RTT) that usually characterizes radio links (e.g., 200 ms).

In literature, this requirement of good performance under noisy links with high RTT is capturing interest and stimulating the birth and growth of specific works. Some of the most significant proposals are TAROC [17] and EPIC [18] [19].

TAROC (TCP-Aware ROBust header Compression) is a robust and efficient header compression scheme for TCP/IP streams whose main goal is the limitation of error propagation through the synergy between a powerful encoding scheme for header fields, named $W$-LSB encoding (well defined in [20]), and the well-known TCP congestion window tracking [21].

EPIC (Efficient Protocol Independent Compression) is an algorithm that can be used to produce an optimum set of compressed header formats for an arbitrary protocol stack and a chosen level of robustness: through the analysis of the possible values and variations of the compressed header fields and the use of a Huffman encoding variant [22], EPIC derives a set of compressed header formats stored in the form of a tree that can be used to quickly compress and decompress headers $^1$.

The ROHC scheme proposed by IETF [23] takes a great step forward with respect to all previous header compression schemes. The success of ROHC is due to the synergy between sophisticated encoding methods (such as the previous mentioned $W$-LSB encoding) and the wide availability of packet formats needed for sending compressed information. ROHC elasticity, however, derives from the concept of the compression profile, that is the set of

$^1$It can be shown that the set of the compressed headers is minimal (average header size).

Euro. Transac. Telecomms. 2004; 0:0–0
logical rules for the compression of a particular kind of packet stream and the set of syntax rules for the interpretation of the compressed packets on the decompressor side.

To be more specific, the ROHC original scheme [20] specifies four compression profiles, one for each "kind" of stream: Profile 1 for RTP/UDP/IP streams (real time services), Profile 2 for UDP/IP streams, Profile 3 for ESP/IP streams and Profile 0 for all other kinds of streams. The main goal of the ROHC workgroup was the introduction of a good compression profile for real time streams in wireless networks, so all efforts were directed to the development of Profile 1. On the contrary, the compression of TCP streams was initially managed with Profile 0, which appears to be totally inefficient, since it only adds an extra header field for the identification of different streams over the same channel‡, without any compression. The result is that a wide range of applications (Telnet, FTP, SMTP, etc.) are heavily penalized§.

Only in more recent times (from the beginning of 2002), the work of IETF led to a specific compression profile for TCP flows [24] (currently still incomplete) within the ROHC platform. Such a proposal follows the guidelines of [20], but it also presents some remarkable differences and still unspecified features, such as a different state machine for the compressor, the introduction of a new header field (called MSN, Master Sequence Number) to help the compressor when receiving feedback packet from the decompressor, and some mechanisms (such as Context Replication) to better treat short-lived TCP transfers.

In this paper, a header compression scheme for TCP streams, as a specific header compression profile integrable in the ROHC platform and compliant with IETF requirements for ROHC TCP/IP header compression [25], is proposed and analyzed by extensive simulation experiments exploiting the Network Simulator (ns) [26]¶. To this end, this new compression profile, developed in parallel with IETF proposal [24], includes all features needed to implement a valid simulation test. In fact, many simulation experiments have been done by considering a simple case of TCP/IPv4 header, without optional fields (such as SACK, Timestamp, ECN, etc.) and assuming classical variations of all header fields, disregarding some details proposed by IETF in the complete scenario. This work has to be viewed as an initial but concrete step, within a more complex and general workplan, towards the realization of a complete and fully functional header compression scheme, based on ROHC guidelines, for TCP/IP streams. Nevertheless, our simulation results, illustrated in the last part of the paper, capture the essential aspects of the problem and appear to us as a good starting-point for further refinements. Our profile is also utilizable, for an extra optimization, with the EPIC scheme.

The rest of this paper is structured as follows; Section 2 illustrates the general characteristics of the ROHC scheme as proposed by IETF [20]. Section 3 describes the fundamental features of our scheme for TCP/IPv4 header compression, while the results of our simulations are presented in Section 4. Finally, Section 5 presents some conclusions and hints for future research.

‡The "CID field", as described later.
§The necessity and importance of TCP/IP header compression on low- or medium-speed links are discussed in [11] and [24]
¶The source code for the compressor and decompressor units is implemented and integrated in ns-2.
2. Basic Concepts about Header Compression with ROHC

To perform header compression, two new functional units, the header compressor and the header decompressor, are needed in the TCP/IP architecture: these units are positioned just under the network layer of mobile units and network nodes and they are transparent to the upper layers. Figure 1 shows the case of a mobile unit-core network link of a 3G wireless system.

The basic idea of each header compression scheme is the observation that during the same packet stream most fields of consecutive packets headers are identical, while many other fields exhibit constant and well-known variations. The ROHC compression scheme uses this observation to classify all header fields into three groups:

- **STATIC**: fields that do not change among the headers of the packets belonging to the same connection (for example source and destination IP addresses); they can be transmitted only at the start of the connection;
- **DYNAMIC**: fields that may change by variable quantities among the headers of consecutive packets belonging to the same connection; they have to be transmitted one or more times, depending on their variations during the connection, by using convenient encoding methods;
- **INFERRABLE**: redundant fields that can be determined through the knowledge of either STATIC and DYNAMIC fields or other information provided by lower levels; they can never be transmitted.

Figure 1. Header compressor/decompressor placing in the TCP/IP platform.

The set of static and dynamic fields of the protocol suite used for the considered service (application) constitutes the context, i.e., the useful information for the compression and decompression process. The ROHC header compression scheme, as illustrated in figure 2, maintains a context for every flow on both the compressor and the decompressor sides to compress and decompress correctly.

Every context uses a compression profile, which is the set of logical rules for the compression of a particular kind of packet stream and the set of syntax rules for the

*Euro. Transac. Telecomms.*, 2004; 0:0-0
interpretation of the compressed packets on the decompressor side.

Since there can be more than one packet stream on the same link, every context is identified by a Context IDentifier (CID). Both the CID and the compression profile associated with a specific context are notified to the decompressor by appropriate packets at the beginning of the connection.

During this initial phase of connection the compression level is very low because the compressor has to send uncompressed information to properly set the context at the decompressor. Subsequently, the compressor sends only the encoded variations of dynamic fields with appropriate compressed formats.

The decompressor maintains a memory of the context for the whole connection time in a CID indexed table, refreshed for any packet received and then decompressed. Normally, the decompressor keeps synchronized with the compressor to correctly perform its functions. But in some conditions, the decompressor context may be out of sync, due, for example, to packet losses. This inconsistency may cause incorrect decompression of successive packets that could be dropped. This effect, called "error propagation", lasts until the contexts are re-synchronized and brings down connection performance, especially on relatively high BER links such as wireless channels.

2.1. Compression States

Header compression with ROHC can be characterized as an interaction between two state machines, a compressor machine and a decompressor machine.

The functionalities of the ROHC compressor machine are based on three states of compression, each one characterized by a different level of compression: Initialization and Refresh (IR), First Order (FO) and Second Order (SO).

At the beginning of the stream, the compressor starts in the IR state with the transmission of all static and dynamic fields in uncompressed form plus some additional information, allowing the decompressor the correct context acquisition. When the compressor is confident that the decompressor has acquired at least the static part of the context, it enters the FO state, where only the encoded variations of the dynamic fields (changed with respect to the header of the previous packet) are transmitted. The compressor reaches the SO state, which corresponds to the highest level of compression, only when the stream becomes "regular" (this happens when all dynamic field changes are fully predictable by knowledge of a "leader header field"), and

\[\text{e.g., "RTP sequence number" in RTP/UDP/IP streams.}\]
it is confident that the decompressor has acquired all the parameters to bind the leader field
with the rest of the dynamic fields. In such a way, in SO state only the encoded variations of
the leader field are transmitted, so that header size can be very small ($3 \div 5$ bytes for TCP/IP
compression with our profile, $1 \div 3$ bytes for RTP/UDP/IP compression with ROHC profile 1
[20]).

When one or more dynamic fields change in an irregular manner, the compressor leaves
SO state and enters FO state, where it again sends the leader field changes but also the
variations, encoded W-LSB [20], of the dynamic fields that exhibit irregularity. Only if
regularity is resumed can the compressor return to SO state; otherwise, if irregularity persists
the compressor has to refresh the context of the decompressor, either remaining in FO state
but reducing compression ratio (in order to send new uncompressed information) or coming
back to IR state.

2.2. Decompression States

The decompressor machine also has three states of decompression, depending on which parts
of the context have been correctly established, as shown in figure 3: No Context (NC), Static
Context (SC) and Full Context (FC).

Figure 3. State machines for the decompressor and for the compressor in all operative modes.

Initially, while working in the No Context state, the decompressor has not yet successfully
decompressed a packet. Once a packet has been decompressed correctly (for example, upon
reception of an initialization packet with all static and dynamic information), the decompressor
can switch directly to the Full Context state. After repeated failures ($k_1$ out of $n_1$, where $k_1$
and $n_1$ are implementation parameters), the decompressor will revert to the Static Context
state; there, the reception of any packet sent by the compressor in the FO state is generally
sufficient to enable transition to the Full Context state again. Only when decompression of
several packets ($k_2$ out of $n_2$, where $k_2$ and $n_2$ are implementation parameters) sent in the
FO state fails in the Static Context state, will the decompressor go back to the No Context
state, waiting for another initialization packet.
2.3. Modes of Operation

The previously described ROHC compression mechanism is also regulated by three different modes of operation: Unidirectional, Bi-directional Optimistic and Bi-directional Reliable. Every mode controls the state transition logic and the detailed actions to perform in each state. Figure 3 shows the state machines for the compressor in all operative modes, while the state machine for the decompressor is the same in all modes (figure 3), even if the choice of implementation parameters depends on the current mode of operation.

In the Unidirectional mode (U-mode), packets are sent only from compressor to decompressor. This mode makes ROHC useful over links where a return path from decompressor to compressor is unavailable or undesirable (i.e., reliable links).

As shown in figure 3, in U-mode, transitions between compressor states are performed according to three mechanisms:

- **optimistic approach principle**: the compressor can transit to a higher compression state only when it is fairly confident that the decompressor has received enough information to correctly decompress packets sent according to the higher compression state; the compressor normally obtains its confidence in decompressor status by sending several packets with the same information according to the lower compression state;

- **timeouts**: since there is no feedback between decompressor and compressor, there is always a possibility of failure since the decompressor may not have received sufficient information for correct decompression; therefore, the compressor must periodically switch to lower compression states;

- **irregularities**: the compressor must also immediately revert to the lower state when the header to be compressed does not conform to the established pattern.

Due to the periodic refreshes and the lack of feedback to initiate error recovery, compression in the Unidirectional mode is less efficient and has a slightly higher probability of loss propagation compared to any of the Bidirectional modes.

The Bidirectional Optimistic mode (O-mode) is different from the Unidirectional mode because a feedback channel is used by the decompressor to send error recovery requests (i.e., NACK or STATIC_NACK) and acknowledgments (i.e., ACK) of significant context updates; the optimistic approach principle and the transitions needed for updates work in the same way as in the U-mode but there are no periodic refreshes of the context. The use of ACK for updating packets is an optimization of the optimistic approach principle: in fact, upon reception of an ACK for an updating packet, the compressor knows that the decompressor has received the acknowledged packet and the transition to a higher compression state can be carried out immediately. Thus, O-mode aims at maximizing compression efficiency through a sparse usage of the feedback channel.

The Bidirectional Reliable mode (R-mode) has two fundamental differences from the previous two modes: a more intensive use of the feedback channel and a stricter logic at both compressor and decompressor that prevents loss of context synchronization. Feedback is sent to acknowledge all context updates, thus the upwards transitions are determined by the secure reference principle: the compressor bases its confidence only on acknowledgments received from the decompressor. Downward transitions are triggered by the need for updates or by negative acknowledgments (NACKs and STATIC_NACKs), as described in O-mode.
R-mode aims at maximizing robustness against losses and damage propagation, even under header loss/error burst conditions, through a reduction of compression ratio.

3. Proposed Scheme

In this section, we summarize the fundamental features of our header compression scheme for the compression of a **TCP stream** unidirectional stream [27] that utilizes **IP version 4** [28] at network layer. In our proposals, for the sake of simplicity, we have only considered compression of unidirectional streams, although the proposed scheme is suitable for accommodating compression of bidirectional streams. The proposed scheme is based on the same general principles of the ROHC profile 1 proposal (as described in the previous section and in details in [18]), but it introduces some variants that are necessary due to the wide differences between UDP flows (profile 1) and TCP flows (our profile). Further details about the proposed scheme can be found in [29], [30], [31], and [32].

3.1. TCP streams

The UDP/IP [33] and RTP/UDP/IP [14] are connectionless and unreliable streams: the UDP source sends the datagram to the IP layer without any guarantee of their delivery to the destination host. For these types of stream, ROHC profiles 1 and 2 are able to realize a robust and efficient header compression.

The TCP/IP, however, are connection oriented streams that implement a bi-directional reliable connection between two end-points (i.e., client and server): this means that before data exchange, the protocol must establish a connection, regulated by the three-way handshake policy [4], after which the TCP must guarantee the delivery, without error, of all packets to the client.

As is widely known, the TCP divides the data stream generated by the server application into *segments*. Every *packet*, at the IP layer, contains a segment as payload and the sequence number of the first byte of the carried segment. On the client side, for any segment received, the TCP sends an acknowledgement segment that contains the sequence number of the first byte of the segment that the client expects to receive. On the server side, before sending new segments, the TCP expects to receive ACK for all sent segments until the expiry of a timeout after which it supposes the loss of the segment and retransmits it. The management of the timeout length and retransmission policies are described in the congestion control protocol [21].

Thus, a TCP unidirectional connection can be described by combining two streams: *data stream* (server to client) and *acknowledgement stream* (client to server). Our scheme performs a separate and independent compression of these two streams. Each one is characterized by the presence of a "**key field**" that the compressor has to monitor with the aim of establishing whether the stream is regular or not (see below for a better description of this concept). These key fields are **TCP Sequence Number** (TCP SN) for the data stream and **ACK Number** (ACK SN) for the ACK stream**.

**These fields have the same behaviour of RTP Timestamp field in ROHC profile 1, so that we can extend...**

Euro. Transac. Telecomms. 2004; 0:0–0
For the compression of these two streams, our scheme introduces the following fundamental innovations with respect to the guidelines of ROHC profile 1:

- specific classification of TCP header fields, obtained with the same approach adopted by [20];
- new formats for the transmission of ROHC headers, fully compatible with the hierarchical structure of the ROHC compressed format;
- appropriate rules for the context update management at the decompressor, that include the adoption of specific mechanisms for the context repair which are more suitable for TCP streams.

3.2. ROHC Sequence Number

Each ROHC compression profile needs to use a dynamic field, called “leader field”, to achieve the maximum compression ratio. The leader field is characterized by constant variations between consecutive packets of the same stream; the compressor monitors all header field variations during the streaming; when these variations are fully predictable only by the knowledge of the leader field, it enters the SO state (“regular flow”) and sends only the variation of the leader field, encoded W-LSB, with high benefit on the overhead and on the robustness level††. When the decompressor receives a compressed packet with only the leader field variation, it uses this information to evaluate the values of all other fields by using specific decompression functions.

In the case of the TCP stream, a dynamic header field with such good characteristics of regularity does not exist so we introduced an ad hoc field called ROHC Sequence Number (RSN), 2 bytes long. The RSN field is generated by the compressor and for the syntax of the compressed formats, a new extra dynamic field of the TCP header it is considered (see below). For every new TCP stream, the compressor sets RSN to a random initial value and transmits it uncompressed. Subsequently, the RSN field is increased by one for every new transmitted packet.

It is important to note that the features of RSN are slightly different from those of the MSN field that is used by the ROHC proposal for TCP header compression [24].

3.3. Classification and management of TCP header fields

When designing a header compression scheme, it is of fundamental importance to understand the behavior of the header fields in detail. The classification of IP header fields is made in [20]. Here we propose our classification of TCP header fields, in compliance with ROHC classification criteria.

First, we have a general classification (table I) on the basis of stable knowledge and assumptions; after this, a specific classification (table II) takes into account the characteristics of CHANGING fields.

††W-LSB encoding guarantees the correct decompression also in the presence of consecutive packet losses.

Euro. Transac. Telecomms. 2004; 0:0–0
Table I. Classification of TCP header fields.

On the basis of these classifications the last column of table 1 indicates how TCP header fields should be handled by the header compression scheme to optimize compression and functionality: our rules perfectly follow the guidelines of section A.3 of [20].

TCP checksum and Urgent Pointer are the only two totally irregular dynamic fields and must be transmitted as they are. Moreover, the TCP checksum field is present in every TCP segment so that it must be present in all compressed packets.

Table II. Classification of CHANGING TCP header fields according to their expected change patterns.

For the other TCP dynamic fields (sequence number, acknowledgement number and window size), including the extra field RSN TCP, we suggest exploiting the W-LSB encoding to achieve the double goal of a more efficient header compression and a greater robustness against packet losses.

Moreover, for a more efficient compression, data stream (server to client) and ACK stream (client to server) are compressed separately. In fact, the fields SN TCP (TCP sequence number) for the data stream and ACK TCP (TCP acknowledgement number) for the ACK stream have the same behavior as TS RTP field (RTP Timestamp) in ROHC profile 1 [20], so that we can extend some aspects of the ROHC profile 1 to the TCP stream by the following logical
association:

- **Acknowledgement number** → TS RTP for the ACK stream
- **Sequence number** → TS RTP for the data stream

In doing this we can also easily extend, with suitable modifications, the syntax for ROHC profile 1 compressed formats, realizing the full integration of TCP profile in the general ROHC scheme. This issue is illustrated in the following paragraph.

### 3.4. Packet formats

In defining packet formats in our scheme, we kept ROHC guidelines [20] in mind and profile 1 compressed formats [20], with the aim of making the integration of our proposal suitable for the general ROHC framework‡‡. Obviously, in order to consider the differences between RTP/UDP/IP header (profile 1) and TCP/IP header (our profile) as described in the previous paragraph, we introduced some variations on fields that are carried by every format and on rules for sending various formats. Moreover, our choice of compressed packet formats takes into account all possible "combinations" of protocols at network layer (IPv4, IPv6, IPv4/IPv4, IPv6/IPv6, IPv4/IPv6 and IPv6/IPv4), even if our work focuses only on the IPv4 case.

In this paragraph the formats of our scheme are presented and we focus attention on the differences between these formats and those used in the ROHC profile 1 (for more details see [20]). The principal differences can be summarized as follows:

- the **Profile** field value in IR and IR-DYN packets is always equal to 4, in order to identify the specific profile for TCP/IP header compression;
- the **static chain** and the **dynamic chain** finish, respectively, with the static and dynamic set of TCP header fields;
- the RTP SN (RTP Sequence Number) field is replaced by the **Roch Sequence Number (RSN) field** that is the leader field for TCP streams;
- the RTP TS (RTP Timestamp) field is replaced by the **ACK SN field** in ACK stream and with the **TCP SN field** in data stream, that are the "key fields" for TCP streams;
- the M flag (RTP marker bit) is replaced by the **X0 flag**. If the X0 flag is set, there is an **extra extension field**, named X0, that appears necessary in order to support the efficient transmission of TCP Window Size and TCP Urgent Pointer (transmitted "as-is") and/or to extend the range for the transmission of the SN field. The extension is allocated between the base compressed format and the general extension format (type 0,1,2 or 3), if present;
- in the **general extension format**, the **R-TS flag** is replaced by the **TCP flag** (if the TCP flag is set, the extension field carries the ACK or SN field for data stream or ack stream) while the **RTP flag** is changed with the **TCP flag** (if set, there is an extra field TCP header field flags).

All defined formats carry the CID field (0 ÷ 16 bits), the RSN field (4 ÷ 16 bits) and the TCP Checksum field (2 bytes), since these fields have to be transmitted with every packet in each stream. This means that the minimum compressed packet with our profile is 3 bytes long.

‡‡The same approach has been followed in [24], but without the definition of compressed packet formats.

*Euro. Transac. Telecomms.* 2004; 0:0–0
The type 0 compressed formats (figure 4) change the SN RTP with the RSN TCP field and present an extra field, 2 bytes long, for sending the TCP checksum.

The type 1 compressed formats (figure 5) present various changes depending on the unidirectional stream which they refer to. In fact, as reported previously, two different fields, the sequence number for data stream and the acknowledgement number for ack stream, have the same behaviour as the TS RTP field in the ROHC profile 1. Thus, in the ROHC type 1 formats it appears reasonable to replace the RTP TS field with the SN field (W-LSB encoded) for TCP data stream and with the ACK field (W-LSB encoded) for the TCP ack stream.

Moreover, we introduce (figure 6) an extra extension field, named X0, to allow the efficient

Euro. Transac. Telecomms. 2004; 0:0–0
transmission of the *Push* flag, the *urgent pointer* field and irregular changes of *window size* field to extend the representation range of the ACK (SN TCP) field in the data stream (ack stream).

The **type 2 compressed formats** (figure 8) present changes similar to those of the type 1 formats. Their goal is to keep a high level of compression, even in the presence of a temporary irregularity of the stream. Every type 2 packet can refresh the context to the decompressor side (every packet carries a CRC code).

The characteristics of IP layer set a limit to the format that can be used. There are three discriminatory factors: IP version (IPv4 or IPv6); type, if present, of IP encapsulation (IPvX over IPvX) and type of assignment policy for ID field (only for IPv4). In table III, we show a summary of the allowed formats, remembering that, according to [20], the RND (for the innermost header) and RND2 (for the outer header) flags indicate whether the ID field has to be compressed (RND/RND2 =0) or instead sent as is (RND/RND2=1) in compressed headers.

In formats of type 1 and 2, if X flag is set, a ROHC extension field is positioned after the base format and, if present, after the extension field X0. The general format of ROHC extension field is presented in figure 8.

The last formats are **IR** and **IR-DYN**, used for packets without compression; these formats are identical to those described in [20], with two fundamental differences: first of all, they present the static chain and the *dynamic chain fields* that end with the TCP static and
dynamic chain respectively; secondly, they use some additional bits (from 8 to 32) to carry control information for the decompressor: specifically, these bits allow the compressor to send a new uncompressed value of the delta (see table I) of TCP Sequence Number (for data stream) or ACK Sequence Number (for ACK stream) and to communicate operative mode transitions. The same bits are also used in extension 3 format, in the extra field TCP header field flags.

**Feedback packets format**

With our compression profile the adoption of all three feedback formats used in ROHC profile 1 [20] (ACK, NACK and STATIC NACK) is necessary. The same considerations made for the forward formats have to be extended to the feedback formats. The derived formats are presented in figure 9.
Feedback-1 can be used only to communicate the success of a decompression (ACK). Code field, CID and Add-CID byte have the same meaning as in ROHC [20]. Instead the RSN field carries the W-LSB encoded value of the RSN field, which the ACK refers to.

Feedback-2 has a more flexible use depending on the Acktype value. In figure 10 the format of the feedback option field (Feedback-2) is presented. We have maintained all the ROHC profile 1 options, with the exclusion of type 5 (here used for the requirement of the retransmission of a TCP corrupted segment) and type 6 (reserved for future use).

CID field

To distinguish data streams from ack streams, we suggest the modification of the CID field format, so that a data stream is characterized by the CID most significant bit (msb) equal to 1, instead of an ACK stream by the CID msb equal to 0. In this way, the decompressor, on receiving a packet by the reading of the CID msb, associates a univocal meaning to the ACK or SN TCP field.

Figure 9. Feedback formats.

Figure 10. Feedback option formats.

3.5. Encoding methods

The methods for encoding dynamic header fields [20] are a fundamental tool for achieving either a high compression ratio or a good robustness against packet losses and transmission errors. Our scheme uses three different coding methods:
• **W – LSB encoding** for all fields that have small and regular variations between consecutive packets in the same stream; this method allows us to obtain a good compromise between compression efficiency and robustness. RSN, ACK SN, TCP SN and Window Size fields are W-LSB encoded;

• **offset encoding** for "IPv4 Identification" field (ID): the compressor applies W-LSB encoding not to ID, but to the difference between ID and RSN. In order to exploit the benefits of this encoding method, it is of fundamental importance that the IP source can assign ID with a pure sequential policy or, at least, sequential with random jumps; instead, if ID is assigned with a random policy, it has to be transmitted as-is in every compressed packet;

• **scaled encoding** for TCP Sequence Number or ACK Sequence Number: since each of these fields often has a variation proportional to a constant ($\Delta SN$), the compressor can save many bits in transmitted packets by applying W-LSB encoding to the "scaled value" ($SN_{scaled}$) of the field itself, where $SN = SN_{scaled} \times \Delta SN + SN_{offset}$, and $SN_{offset} = SN \mod \Delta SN$.

For further details about all these coding methods, see [20].

### 3.6. Decompression functions

In presence of regular flow (i.e., variations of SN and ID fields which are fully predictable by the value of RSN), the compressor enters the SO state and transmits only the value of RSN field, W-LSB encoded. When the decompressor receives a packet with only the RSN field, it calculates the values of SN field and ID field through appropriate decompression functions.

These function are mentioned in [20] but not shown; in our profile, we suggest the use of the following expressions:

\[
ID_i = f(RSN_i) = RSN \times \Delta ID + ID_{off}
\]

\[
SN_i = f(RSN_i) = RSN_i \times \Delta SN + SN_{offset}
\]

where $\Delta SN$, $SN_{off}$, $\Delta ID$ and $ID_{off}$ are decompression parameters that belong to the context of the decompressor and $i$ identifies the value of a header field (SN, ID or RSN) for the current received packet. These parameters can be either automatically calculated at the decompressor through the values of ID, SN and RSN fields in the previous correctly received packets (*implicit mode*), or they can be transmitted explicitly from the compressor with appropriate packet formats (*explicit mode*).

### 3.7. Error management

One requirement for header compression scheme in wireless environments, is the robustness, i.e., the ability to manage "errors". In particular, error propagation is the most dangerous effect in a wireless system that uses header compression: this effect implies that the decompressor has to discard many packets (also without errors) when its context is invalidated (due to transmission errors and/or multiple packet losses), until the arrival of one or more uncompressed packets properly sent by the compressor. This effect produces a strong degradation of throughput and so has to be properly addressed.

In the ROHC general scheme [20], this problem is solved through an efficient synergy between four tools (all implemented in our scheme):

---

*Euro. Transac. Telecomms. 2004; 0:0–0*
• **W−LSB encoding** reduces the probability of context invalidation after consecutive packet losses;
• *feedback, timeouts* and *optimistic approach principle* increase the probability of correct synchronization between compressor and decompressor;
• the introduction of a *CRC field* in compressed packets is a reliable error revelation mechanism;
• *local context repair mechanisms* can try to repair the decompressor context directly at the receiving side, without waiting for a specific intervention of the compressor.

The use of W-LSB encoding represents a fundamental tool in preventing synchronization losses and we consider it one of the most important innovations of the ROHC general scheme with respect to previous header compression schemes.

We also introduced another mechanism for the quick recovery of the synchronization between compressor and decompressor based on the typical behavior of TCP sender and receiver. In fact, while the UDP sender does not know anything about what happens at the receiver and sends its datagrams continuously (if there is data ready for transmission), the TCP source has a different behavior: first of all, it is always forced to resend a segment if no corresponding ACK arrives until the expiry of a timeout; moreover, the timeouts prevent the source from sending packets continuously according to the sliding window mechanism. So, in our scheme, every retransmission is taken as an "implicit feedback" for the compressor on the source side: when the compressor observes a non-positive variation of TCP Sequence Number field, it is authorized to think that at least one packet could have been incorrectly decompressed at the receiving node, so it resets its "compression state" and restarts the compression from the IR state\(^\text{§}\). In this way, without any explicit feedback and through an unavoidable reduction of compression ratio (due to the use of an appropriate packet format for the update of decompressor context), synchronization between compressor and decompressor can be recovered more quickly and the effect of "error propagation" can be minimized.

4. Simulations

We conducted several simulation experiments on the *Network Simulator* version 2 [26] to test the performance of our header compression scheme. We used a simple but reasonable network topology of three nodes (figure 11), often used in literature for this kind of simulation [17], that represents a path with a wired reliable link and a wireless noisy link:

• **wireless link**: bandwidth 9.6 kb/s ÷ 384 kb/s, delay 30 ms, BER \(10^{-6} \div 10^{-3}\);
• **wired link**: bandwidth 2 Mb/s, delay 30 ms, reliable.

The wired link is representative of the entire network path over a wired IP network and it is assumed to be perfectly reliable. The propagation delay of this link represents the sum of propagation delay, queuing delay and processing time of each node of the wired network. We assume that the base station does not drop packets due to insufficient buffers.

\(^\text{§}\)The same behaviour is adopted from the compressor at the receiver side, in presence of duplicated ACKs (that is to say consecutive packets with a constant value of TCP ACK Number).
The propagation delay of the wireless link represents the sum of propagation delay, queuing delay and the processing time at the mobile node.

Figure 11. Simulated topology.

It has to be noted that the simulated channels do not correspond to physical channels, but they are the "equivalent channels" under the IP layer, as shown in figure 13. This choice implies that their characteristics (bandwidth, delay, error model) include all logical and physical mechanisms adopted by lower layers (i.e. error correction, framing, and so on).

Figure 12. Detailed architecture of the simulated topology.

A one-way TCP traffic (Reno version [21]) is transmitted from the mobile node (server) to the fixed node (client), with an FTP source at application layer. The TCP header compression module is attached on the wireless link which is the only network resource that has to be optimized from the point of view of header overhead. The bandwidth and delay of the wireless feedback channel are assumed to be the same as those of the wireless WAN link.

We ran a series of simulations on the above topology where the BER over wireless channel varied from $10^{-6}$ to $10^{-3}$. The adopted error model assumes that bit errors are statistically independent: because this is not true for many real links (for example, many radio links are affected by propagation fading or by interference that lasts over many bit times, or many links that use Forward Error Correction have non-uniform bit error distributions that depend on the type of FEC), our calculation (from BER) of the probability of a packet being corrupted can only place an upper limit on the packet loss rate [9].

Since a smaller MTU is better in terms of increasing the chances of successful transmission but is also worse for the header overhead, three different MTUs have been chosen (576 bytes, 296 bytes, 168 bytes).

4.1. Operating assumptions and performance criteria

We made some basic assumptions for our simulations:

- IP version 4 at network layer;
• absence of optional fields (SACK, Timestamp, ECN bits, and so on) either in TCP header or in IP header: this implies that the size of TCP/IP uncompressed header is statically 40 bytes;
• pure sequential assignment of IPv4 Identification field (as also specified in [24];
• use of a virtual channel with piggybacking for the feedback between decompressor and compressor (i.e., feedback packets are transmitted together with forward packets that are directed in the same direction);
• small CIDs (constant contribution of 1 byte to the medium compressed header size);
• no residual errors over packets delivered to the decompressor (i.e., the ”error model” only drops corrupted packets, while random errors introduced during transmission are always detected and/or corrected by lower-layer repair schemes).

Some of these assumptions may appear too extreme for a real scenario, but they have to be viewed within the general aim of this work, which is a preliminary analysis of the achievable performance of the proposed scheme.

In all experiments we compared the performance of normal TCP and TCP with our compression scheme (referred to in the sequel as ”ROHC+”) by using the following performance criteria:

• throughput (in bytes per second) is the ratio between the total amount of payload bytes correctly delivered to the receiver and the time of connection (100 seconds); it corresponds to the speed at which the destination receives user data and thus it is a measure of QoS;
• overhead is the ratio between header bytes and total bytes transmitted by the source over the wireless link; it can be shown that every reduction of the overhead corresponds to an equivalent reduction of the bandwidth consumed for the header transmission (and thus of the bandwidth totally requested by the connection);
• goodput is the ratio between the total amount of payload bytes correctly delivered to the receiver and the total amount of bytes transmitted over the wireless link. It is a global measure of channel efficiency, because it takes into account the amount of correctly delivered packets and also the compression ratio achieved either in the forward stream or in the ACK stream.

4.2. Results

The main target of the developed simulations was the evaluation of two fundamental aspects of the proposed compression scheme: the capability of obtaining good values of throughput even in bad channel conditions and the maximum compression efficiency achievable in good channel conditions. This part of the paper aims to show how these targets, were reached.

Regards the first ”target”, in all simulations there was an important throughput improvement with ROHC+, which demonstrates that our proposal is sufficiently robust against noisy link. Table IV shows the throughput measured in U-mode (no feedback between compressor and decompressor) with different MTUs and with a bandwidth of 384 kb/s over the wireless channel.

In all simulations there was a light but important throughput improvement with ROHC+ that demonstrates that our proposal is sufficiently robust against noise link.

The throughput improvement, especially over bad wireless channels (when other compression schemes normally fail), was an indispensable ”starting-point” for our scheme; following this, we
were interested in the improvements of overhead and goodput, so the subsequent paragraphs are dedicated to these subjects.

4.2.1. Results for Unidirectional Mode  The majority of our simulations involved the use of our compression scheme in Unidirectional mode, since this operative mode is the "weakest" for the behaviour of compressor and decompressor.

In table V the measured overhead and goodput are reported with different MTUs and in different channel conditions considering 384 kb/s over the wireless channel.

The results show that our scheme assures good improvements of overhead and goodput for every value of BER: under good channel conditions, TCP SN and ACK SN fields are sufficiently regular (due to the small number of packet losses) and the compressors on each side of the connection can reach the SO state more frequently and for longer periods; thus we can have an improvement of 80% or even greater; on the contrary, under bad channel conditions, the high number of packet losses implies a great number of retransmissions (at TCP sender) and duplicated ACKs (at TCP receiver), restraining the compressors in their transition towards the SO state and thus reducing the achievable improvements.

| Table V. Overhead% and Goodput% vs BER, for different MTU, with 384 kb/s over wireless. | Euro. Transac. Telecomms. 2004: 0:0–0 |
In figure 13 the behaviour of overhead under different BER conditions is reported considering different values of the wireless bandwidth and assuming an MTU equal to 296 bytes. For the same value of the MTU, figures 14.(a) and 14.(b) show, respectively, the overhead and the goodput measured for a fixed BER of $3 \cdot 10^{-5}$, considering different values of the bandwidth (9.6 kb/s, 14.4 kb/s, 64 kb/s, 96 kb/s, 144 kb/s, 384 kb/s).

Figure 13. Overhead% vs BER, for MTU=296 bytes, with different bandwidth over wireless.

Figure 14. Overhead% and Goodput% vs bandwidth over wireless (MTU=296 bytes; BER=3 \cdot 10^{-5}).

For every value of BER the compression ratio decreases with the bandwidth for two reasons: the number of packets generated by the TCP sender is proportional to the bandwidth and the compressor needs to receive a large number of IP packets at high frequency in order to exploit
the periods in which the flow is regular. In any case, the achieved improvements appear very clear.

An important result is the compression ratio over the reverse channel (ACKs from client to server). In figure 15 we report the medium header size measured either in the forward direction (DATA from server to client) or in the reverse direction, for MTU=296 bytes and 384 kb/s over wireless, with different BER conditions.

Figure 15. Medium header size vs BER conditions, for MTU=296 bytes, with 384 kbit/s over wireless.

It is clear that the compression ratio in the reverse channel is rather worse than in the forward channel. This happens because the behaviour of ACK Sequence Number field in the ACK stream is always less regular than that of the TCP Sequence Number in the data stream while keeping the compressor more frequently in lower compression states. When channel conditions are particularly bad, the compressor never leaves the IR state, in which all fields are transmitted in uncompressed form (headers of 41 ÷ 43 bytes).

4.2.2. Results for Bidirectional modes Final simulations were made with the aim of investigating the features of Bidirectional operative modes (Optimistic and Reliable). Figure 16.(a) contains a summary of measured results (with 384 kb/s over wireless channel), showing goodput and throughput, in all operative modes, with a fixed MTU (296 bytes) and variable BER conditions. For the same simulation, figure 16.(b) shows the medium amount of feedback bytes sent by the decompressor of the data stream, calculated as the ratio between the total amount of feedback bytes and the number of forward compressed packets.

The results lead us to the following conclusions:

- when the channel exhibits bad conditions (BER > $10^{-4}$), the performance of U-mode and O-mode are almost the same and indicate a better goodput with respect to the R-mode; the R-mode lets us obtain a good improvement ($\approx 13\%$) of throughput as a consequence of the intensive use of feedback (more than 3 feedback bytes for each forward packet sent from the compressor to the decompressor with BER=$10^{-3}$); thus, in the R-mode more data is delivered to the final user through a less efficient use of the channel;
on the contrary, under good conditions of the channel, the O-mode allows us to obtain an improvement of both the compression ratio (better goodput) and the final throughput by utilizing the feedback in an "intelligent" way (less than 0.03 feedback bytes for each forward packet with BER $\leq 10^{-5}$).

Figure 16. Results for bidirectional mode.

(a) Goodput and throughput vs BER, for MTU=296 bytes and 384 kb/s over wireless, in all operative modes. (b) Medium amount of feedback bytes vs BER, for O-mode and R-mode.

However, a greater improvement of the goodput in O-mode was expected with respect to the one measured in simulations. The reason for this unexpected result is that the main effect of O-mode is the reduction of the number of packets sent with minimum level of compression (optimization of the optimistic principle approach), but in many situations this number is only limited, as is the corresponding effect. Therefore, we tested the O-mode performance over a more "limited" wireless channel with bandwidth of 64 kb/s and delay of 100 ms. The results obtained are shown in figures 17.(a) and 17.(b), with a MTU equal to 168 bytes.

In both directions, the O-mode produces an improvement of the compression ratio with respect to the U-mode. However, since the ACK stream is more "unstable" (because of the irregularity of ACK Sequence Number field), the "weight" of packets with low compression level is greater for this stream and thus the effect of O-mode appears to be stronger than that found in the forward direction.

5. Conclusions

This paper describes a header compression scheme for TCP/IP streams in a wireless all-IP context, such as future 3G platforms. Our objective has been to define a new compression profile that could be perfectly integrable in the ROHC scheme. For this reason, particular attention has been paid to the description of the compressed packets and state machines, in order to make a real integration of our scheme with ROHC feasible.

Euro. Transac. Telecomms. 2004; 0:0–0
The simulation results reported in this paper, obtained from a simplified but reasonable case of TCP/IPv4 streams, show that our compression scheme is able to provide an efficient use of radio resources with direct benefits on the economic feasibility and on the quality of the service, either with good or bad conditions over wireless links.

In particular, our simulation experiments suggest the following considerations:

- in all cases, the results are directly dependent on the variations of TCP Sequence Number (for the data stream from server to client) and ACK Sequence Number (for the ACK stream from client to server) fields during each stream. Thus it is important to characterize the typical variation patterns of these fields in real networks; more specifically, the compression over the reverse channel is less efficient than that over the forward channel as a consequence of the irregularity of ACK Sequence Number field;
- the performance of the compression scheme seems to be very “promising” in all considered cases: with good channel conditions we obtained a medium header size, in the forward direction, of 5.3 bytes, which is very near to the limit of 4 bytes indicated in literature as the minimum reachable value [17]. On the contrary, under bad channel conditions, the compression ratio is inevitably very low but the phenomenon of error propagation is minimized and this feature appears to be a fundamental requisite of every compression scheme for wireless networks;
- finally, even if the reported simulation results considered IP version 4 at network layer, the proposed scheme can easily be extended to support IPv6. The IPv6 header has some features, such as the large size of static fields, the reduced size of dynamic fields, the absence of a complex field like IPv4 Identification, and so on, that appears particularly suitable for the proposed profile.

Euro. Transac. Telecomms. 2004; 0:0–0
REFERENCES


AUTHORS’ BIOGRAPHIES

Gennaro Boggia received, with honors, the Dr. Eng. Degree in Electronic Engineering in July 1997 and the Ph.D. degree in Electronic Engineering in March 2001, both from the “Politecnico di Bari”, Italy. Since September 2002, he has been with the department of “Elettrotecnica ed Elettronica” at the “Politecnico di Bari”, Italy, where he is currently an Assistant Professor. From May 1999 to December
1999, he was visiting researcher at the CSELT (Centro Studi e Laboratori Telecommunicazioni, presently "TelecomItalia Lab") where he was involved in the study of the Core Network for the next releases of UMTS. His research interests span the fields of Wireless Networking, Cellular Communication, Queueing Networking, Networks Performance Evaluation.

**Pietro Camarda** received a University Degree in Computer Science from the University of Bari, Italy, in 1978. Since 1986, he has been with the department of "Elettrotecnica ed Elettronica" at the "Politecnico di Bari", Italy, where he is currently an Associate Professor of Telecommunications. He was visiting scholar from October 1986 to December 1987 at Computer Science Department of the University of California, Los Angeles. He has been involved in various research areas, mainly: LAN/MAN Architectures and Protocols, Optical Networks, Network Reliability, Cellular Radio Networks, Telecommunication Services.

**Sandro Petrizzelli** received the Dr. Eng. Degree in Electronic Engineering in December 2001 from the "Politecnico di Bari", Italy. Since 2002, he has been with Tecnopolis Csata (Bari, Italy) where he is currently Network Engineer. His work interests span the fields of Wireless Networking, Internetworking and System Integration.

**Vito G. Squeo** received the Dr. Eng. Degree in Electronic Engineering in April 2001 from the "Politecnico di Bari", Italy. Since 2001, he has been with Luxhora IT Company (Bari, Italy) where he is currently CRM Engineer. His work interests span the fields of Wireless Networking, System Integration, CRM Application in Telecommunication System.

Euro. Transac. Telecomms. 2004; 0:0–0