Computer Science and Information Systems 14(2):283 –308 DOI: 10.2298/CSIS151101006T

Performance Evaluation and Implementation of IP and Robust Header Compression Schemes for TCP and UDP Traffic in Static and Dynamic Wireless Contexts

Máté Tömösközi[1,2,3,4], Patrick Seeling[5], Péter Ekler[3], and Frank H.P. Fitzek[4]

[1] acticom GmbH, Am Borsigturm 42, 13507 Berlin, Germany
[2] Department of Electronic Systems, Aalborg University;
Fredrik Bajers Vej 7B, DK 9220 Aalborg, Denmark
[3] Department of Automation and Applied Informatics, Budapest University of Technology and Economics, Hungary; Q. Building, Magyar tudósok krt. 2., HU 1117 Budapest,
[4] Communication Networks Group, Technische Universität Dresden;
Würzburger Straße 35, 01087 Dresden, Germany
[5] Department of Computer Science, Central Michigan University;
Mount Pleasant, MI 48859, USA
mate.tomoskozi@[acticom.de, aut.bme.hu, tu-dresden.de]
pseeling@ieee.org
ekler.peter@aut.bme.hu
frank.fitzek@tu-dresden.de

Abstract. Modern cellular networks utilising the long–term evolution (LTE) set of standards face an ever–increasing demand for mobile data from connected devices. Header compression is commonly employed to minimise the overhead for IP–based cellular network traffic. In this paper, we evaluate the three header compression implementations used by these networks with respect to their potential throughput increase and complexity for different mobile service scenarios over wireless IP networks. Specifically, we consider header compression as defined by (i) IP Header Compression (RFC 2507), (ii) Robust Header Compression version 1 (RFC 3095), and (iii) the recently updated Robust Header Compression version 2 (RFC 5225) with TCP/IP profile (RFC 6846). The contribution of this article is the performance evaluation of IP Header Compression (IPHC) for UDP and TCP, as well as its evaluation in contrast to the Robust Header Compression (RoHC) methods in a comparative overview for real–world mobile scenarios. Our results show that all implementations have great potential for saving bandwidth in IP–based wireless networks, even under varying channel conditions. While both RoHC versions generally provide more reliable results than IPHC, we find that on a unidirectional channel IPHC could perform better. However, if a TCP connection is prone to packet reordering (e.g., by retransmissions), IPHC's performance drops drastically, while RoHC's does not exhibit any significant performance reduction.

Keywords: Robust Header Compression; Mobile multimedia; Cellular networks; Bandwidth savings; Linear regression; Machine Learning

1. Introduction

Mobile data is increasingly transported over the Internet Protocol (IP) in both versions (IPv4, as well as IPv6), especially with the implementations of third generation (3G) and long–term evolution (LTE) networks of the fourth generation (4G). The increased data rates achievable in these newer networks allow for further media convergence on mobile devices, such as video conferencing or video streaming. Wireless systems in general, but especially cellular networks, in turn face increasing data consumption demands from mobile consumers – a trend that will likely continue in the foreseeable future, as indicated by, e.g., [36]. In addition to media consumption, there is a huge demand for data and voice transmissions (e.g., VoLTE, Voice over IP) for mobile devices. Similarly, the proliferation of social networks and similar applications are requiring many signalling messages for mobile scenarios. Because of the widespread adoption of LTE in cellular networks worldwide next to the IP–based nature of services employed within these networks, a large disparity exists between the common packet payloads and the headers required for transmissions. Header compression methods can be applied to reduce the encapsulation overheads and, in turn, these compression methods can save significant amounts of bandwidth.

As a result of a slower pace of capacity building through the expansion of the infrastructure, which is partially driven by the high costs of associated investments, the capacity increase due to the optimisation of mobile communications in cellular networks has attracted a great deal of research and practical implementations in the past. The motivation behind this paper is to compare both Robust Header Compression versions and IP Header Compression commonly encountered in today's cellular networks. Additionally, our evaluation provides a comparative analysis by employing working implementations of these schemes for the first time.

All IP packets carry the typical protocol encapsulation overheads of one or more protocols from the IP protocol stack, independent from actual payload sizes. For mobile multimedia settings, a common protocol combination is RTP/UDP/IP; these protocols specifically account for an encapsulation overhead of 40 bytes with IPv4 and 60 bytes with IPv6 for each individual packet carried over the network. A large portion of IP and UDP packets (such as source and destination addresses or port numbers, etc.) could be omitted for most of the packet traffic as they stay constant throughout the transmission. Other fields such as the IP Identification field or TCP Sequence Numbers can be derived from a single Master Sequence Number.

One major approach to the reduction of data that is sent over wireless networks is header compression, which reduces the protocol encapsulation overhead between, e.g., wireless base stations and mobile clients. In the past, research and implementation efforts targeted a reduction of these protocol overheads, as the effect of reduced transmission sizes of individual network packets quickly multiplies. The overall approach for compression of the protocol headers is based on a compressor/decompressor concept, whereby both are located between the data link layer and the network layer on a sender/receiver pair's protocol stack implementations. The reduction in protocol overheads, typically comprised of RTP/UDP/IP protocol headers in, e.g., a Voice over IP (VoIP) scenario, exploits the redundancy commonly encountered (i) among the different headers of individual packets and (ii) between consecutive packets belonging to the same IP flow.

Several header compression approaches were proposed over the years, with IP Header Compression (IPHC) and Robust Header Compression (RoHC) in its original version (RoHCv1) and its recent update (RoHCv2) representing popular implementation choices today. While these different design alternatives are available from network and device provider points of view, little comparative evaluation has been performed to assess the advantages for each scheme under realistic mobile service conditions. In this article, we perform and show this comparison – in the context of UDP and TCP traffic – in order to fill the gap in the existing research. We refer to [42] for our evaluation of RTP compression efficiency comparison between RoHCv1 and RoHCv2.

The rest of this paper is structured as follows. In the succeeding section, we briefly note previous research that has been performed in this area. We perform a review of the underlying general compression mechanisms in Section 3. We continue in Section 4 with a description of the overall experimental configuration and the performance metrics utilised to evaluate compression performance and complexity. In Section 5 we determine the efficiency of RoHCv1, RoHCv2 and IPHC for UDP–based audio streams in an error–free channel setup. Subsequently, we measure the performance differences between RoHC and IPHC for TCP scenarios in Section 6. In Sections 7 and 8 we provide our evaluation of UDP and TCP over error–prone channels, respectively. Section 9 concludes the article and proposes further research directions.

2. Related Work

In this section we provide a brief historic introduction to header compression before we review current related works. The first IP header compression scheme was the *Compressed Transport Control Protocol* (CTCP or VJHC). It was proposed by Van Jacobson [18] and only considers the TCP protocol. CTCP combines TCP and IP headers together for better results and lower complexity. The compression algorithm itself employs *delta coding*, which refers to differences between two packets. The advantage of this approach is the high compression ratio. Unfortunately, this method is also very susceptible to bit errors, which results in the dropping of numerous packets following an erroneous one by the receiver. CTCP also relies on the lower and higher level protocols' protection schemes as there is no built–in error detection of its own.

An improvement to this was introduced by *Perkins* in [31]. The delta coding for the neighbouring packets is replaced by a reference frame, much like modern video compressions. This results in better tolerance to errors compared to CTCP, albeit produces less compression gain. An enhanced version of this approach by *Calveras* ([5] and [6]) employs a dynamic frame length scheme as a function of the channel state. However, both of these approaches suffer from desynchronisation when the first (uncompressed) packet is lost, which results in the corruption of all other packets in the same frame. A proposed improvement is available by Rossi ([37] and [38]).

As the next step, the compression of RTP was addressed with the development of *Compressed Real Time Protocol* (CRTP) [7]. *RObust Checksum–based COmpression* [40] is a refinement of CRTP (also called ROCCO), which improves the header compression performance for highly error–prone links and long round–trip times. Similarly, *Enhanced Compressed RTP* (ECRTP) [8] is a refinement of CRTP.

Robust Header Compression has built on these predecessors and version 1 of RoHC was introduced in [3] and was created around the concept of extensibility with various profiles that were added later (see, e.g., IP [21], UDP-Lite [28] and TCP profiles [30]). However, version 2 of Robust Header Compression, defined in [29], chooses simplicity in design over extensibility.

We provide an overview of compression profiles defined for RoHC version 1 and version 2 in Table 1. We note that some of these profiles are optional and extend the core RTP/UDP/IP compression with further protocols or protocol combinations. The same is true for IPHC, albeit the RFC (see [18]) does not explicitly define profiles. Instead, it separates UDP and TCP compression and provides an implementation hook for CRTP compression.

However, prior evaluations focused only on RTP and mostly considered RoHCv1. Articles made with the same implementation focused, e.g., on the impact of RoHC on media performances have found that header compression cuts the required bandwidth in half for voice transmissions (GSM) and improves the overall voice quality (see [35] and [14]). Later, the authors additionally discovered that video quality can be enhanced as well (see [39], [14]).

Furthermore, RoHCv1 and RoHCv2 performances were evaluated for RTP in [42] and it was shown that both versions perform equally well for Voice over IP transmissions, albeit RoHCv2's gain is slightly better by 5–10 %. [42] also shows that while RoHCv2 uses more complexity on the compressor side, it is much faster during decompression when compared to RoHCv1.

Table 1. RoHC compression profiles with relevant RFCs and years of effect.

Profile identifier	**Compressed protocols**	**v1 RFC (date)**	**v2 RFC (date)**
0x0000	uncompressed	3095 (2001)	
0x0001 or 0x0101	RTP/UDP/IP	3095 (2001)	5525 (2008)
0x0002 or 0x0102	UDP/IP	3095 (2001)	5525 (2008)
0x0003 or 0x0103	ESP/IP	3095 (2001)	5525 (2008)
0x0004 or 0x0104	IP	3843 (2004)	5525 (2008)
0x0006	TCP/IP	4996 (2007)	
0x0007 or 0x0107	RTP/UDP–Lite/IP	4019 (2005)	5525 (2008)
0x0008 or 0x0108	UDP–Lite/IP	4019 (2005)	5525 (2008)

Research into the application of header compression schemes to multi–hop (ad–hoc) networks have shown that RoHC and IPHC perform reasonably well in such scenarios (see [12], [11], [1]). In [20] it was shown that an end–to–end compression scheme could reduce the delay in time–critical systems, which is a downside of using point–to–point compression techniques like IPHC and RoHC.

An evaluation of IPHC with 6LoWPAN can be found in [24]. For the evaluation of RoHCv1 in WiMAX networks, we refer to, e.g., [46], which points out that the compression performance is better in optimistic mode. In [25] the authors investigate the reliable and optimistic modes and determine that there is no significant gain in voice quality with tighter compressor/decompressor coupling, while in [15], the authors report that around 23 % gain is attained in unidirectional mode only for medium or better voice quality. Similarly, [17] focuses on the behaviour of RoHCv1 in unidirectional mode on lossy links. It was described in [19] and [13] that the RoHCv1 compression savings heavily depend on the compressor's mode state. The non–stateless nature of the compression also poses a potential security risk, which, however, could be mitigated by encryption and authentication techniques as highlighted in [10].

Both [16] and [45] successfully apply RoHCv1 to Aeronautical Networking and [2] to IP tunnel compression, while [27], [23], [33] and [22] propose aggregation in order to improve header compression in mesh networks among other uses. [34] employs header compression to optimise P2P–TV transmission by up to 35 %.

Furthermore, our initial evaluation of Robust Header Compression version 2's energy consumption can be found in [44]. We show that the potential power drain on modern mobile devices is not likely to increase when employing header compression, and there are also indications of possible energy savings via diminished network interface activity. RoHC would therefore benefit the next generation networks of 5G for, e.g., Massive Machine Type Communications (MMTC), since it would require very large number of small and power–constrained devices that need very little outside (human) interventions. For the sake of ensuring long–lasting deployment, energy saving becomes a key element during the design of IoT services and devices.

Similarly, in order to avoid the large number of connections for the MMTC and Vehicle–to–vehicle communications (V2V) for the dissemination of various data (e.g., signalling, status, safety, sensory information) broadcasting could provide a cost and energy efficient alternative. We show in [43] that RoHC can also function in an 1:n setup with the enforcement of certain constraints on various header field behaviours.

Our present research efforts also try to address the issues in highly dynamic networks – such as the above mentioned – by enabling current compressor implementations to configure themselves online, thereby making the compression adaptable to changing channel conditions and various network stream characteristics. For an overview of this topic, we refer to [41].

In the following section we review the main compression approaches inherent to RoHC, and refer to [26] and [9] for a more general overview of RoHC compression.

3. Compression Mechanism

Header compression schemes use states and contexts to maintain knowledge of compressed streams and compressed packet types that can be sent over a link (compressor) or interpreted by the receiver (decompressor). The states can be thought of as a finite state machine structure that is present in both versions (although the actual implementation details are left to the implementer). In this section we describe the different states and interplays in an overview.

3.1. Compression States

The 3 states of the RoHC compressor, illustrated in Figure 1, correspond to: (i) *Initialisation and Refresh state* (IR), where the compressor establishes a new or reinitialises an already existing context; (ii) *First Order state* (FO), in which the compressor synchronises the dynamically changing fields; and (iii) *Second Order state* (SO), where optimal compression is achieved and only the most relevant data is sent.

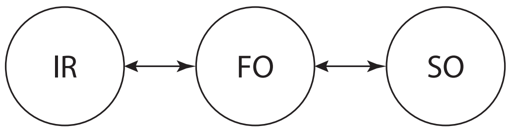

Fig. 1. Compressor finite machine structure with *Initialisation and Refresh* (IR), *First Order* (FO), and *Second Order* (SO) states.

The ROHC decompressor uses a similar approach, which we illustrate in Figure 2. Here the states are: (i) the *No Context state* (NC) representing a decompressor which does not have an established or valid context for the session; (ii) the *Repair Context state* (RC), in which the context is (partially) corrupted and smaller packet types have to be discarded; and (iii) the *Full Context state* (FC), where all packet types can be interpreted.

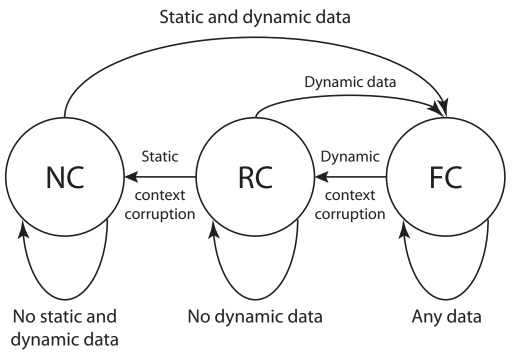

Fig. 2. Decompressor finite machine structure with *No Context* (NC), *Repair Context* (RC), and *Full Context* (FC) states.

Ideally, after the initial context is established, the compressor and decompressor always remain in the third states (i.e., SO or FC). If a desynchronisation occurs, the machine state must transit to a lower one (e.g., FO or RC). If the state is reduced further to the first one, the whole context must be updated before compression can recommence. However, this can be quite undesirable, as the compressor has to transmit larger packets, which are close to or even longer than the original ones. Therefore, this scheme provides a "middle" state (FO, RC), which is able to resume compression with only a partial context refresh.

In IPHC, on the other hand, no explicit compression or decompression states have been defined. However, the decompressor can send a CONTEXT_STATE packet to notify its compressor counterpart when one or more of its contexts have been invalidated and force a refresh.

3.2. Contexts

To achieve better compression ratio, the compressor organises the incoming packets into different contexts according to their characteristics. These characteristics are defined by the *static* (RoHC) or *DEF* fields (IPHC) that never or rarely change during the transmission's lifetime. A good example for one of such fields is the IP source and destination addresses present in the IP header.

By separating the packets according to these fields, the compressor has to send only the "dynamic" data found in the headers. This "dynamic data" – also referred to as *dynamic* and *irregular* fields in RoHC or *DELTA* and *RANDOM* fields in IPHC – usually changes from packet to packet but in a well defined manner. For example, the *IPv4 Identification* number increments by 1 for every packet that follows. This behaviour can be exploited by transmitting an initial value and storing it in the decompressor context. Later, this initial value can be updated according to the difference (delta) between the stored and the new value.

These contexts are bound tightly to the compressor (decompressor) states described in the previous subsection. If a context becomes corrupted or loses synchronisation with the compressor, only that specific context needs to be repaired or reinitialized. Exactly for this reason do the feedback packets contain the context identifier as well.

3.3. RoHCv1 Modes

A unique feature in RoHCv1 is the presence of certain operational modes. These modes govern how the compressor handles context corruption using feedback and pre–emptive context refreshes. In contrast to version 1, the new version of RoHC and IPHC do not explicitly specify these modes. However, similar behaviours can be achieved with the right configuration of the entities (although transitions during run–time are not explicitly supported according to the RoHCv2 RFC).

In *unidirectional mode* there are no feedback packets sent back to the compressor. By supporting this mode, RoHC can be run on links without uplink channels. However, in the absence of feedback there is no way for the compressor to ascertain whether a compressed packet was successfully decompressed on the receiver side. To account for this, the compressor can use the *optimistic approach*, which periodically repeats context initialisation and sends a changed reference value multiple times.

The *bidirectional optimistic mode* uses feedback packets, that are sent from the decompressor to the compressor, in order to accelerate state transitions at the compressor and to avoid the periodic fallbacks to the first and second states. Due to the mostly weak CRC protection, this mode is still relatively prone to context damage and therefore utilises the *optimistic approach* as well. Lastly, the *bidirectional reliable mode* uses a more powerful CRC protection and a very tight coupling between the two endpoints by relying heavily on feedback received from the decompressor.

3.4. Compressed Packet Types

All compressed packets contain a context identifier (CId). The context identifier can either be 0, 1 or 2 bytes long for RoHC and 1 or 2 bytes for IPHC. In RoHC's small–CId mode, a 0[1] or 1 byte long CId field is at the start of a packet, which is followed – in most

[1] A CId of 0 does not need to be transmitted in the compressed packet when using Small-CIds.

compressed packets – by the first octet of the compressed packet type. With large CIds, after the aforementioned first octet is the 1 or 2 bytes long CId information[2].

Following the CId information of the first octet is the compressed packet–type dependent field. In RoHC, after these informational bytes, the last bytes of the packet follow. These contain the various compressed chains (if any). In case of an *IR* packet, this means the *static* and *dynamic chain*s, for an *IR–Dyn* or *co_repair* packet type, only the *dynamic chain* is contained, whereas for the remaining packet types the *irregular chain* is included.

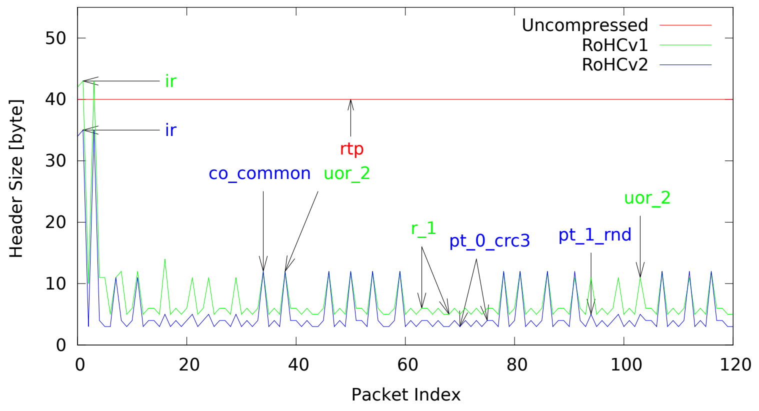

Fig. 3. Header sizes during the compression of an RTP stream with RoHCv1 and RoHCv2. The annotations show the various compressed packet types and the corresponding original headers (RTP/UDP/IPv4). The larger compressed packets (e.g., *IR*, *co_common*, *uor_2*) indicate the transmission of state persistent data, which is crucial for the successful decompression of later packets.

RoHCv1's packets do not contain any chains except for the two *IR*s. Aside from these two, the compressed packets are defined by mode–type–property combinations. The mode identifies the mode in which the compression is currently working. The type basically states how much information is contained in a packet and the optional property extends the basic packet with some additional data (for example *CRC*, *Timestamp* or *IP Id*). For RoHCv1, there are only *static* and *dynamic chains* defined. These are used to transmit the (initial) values of constant (e.g., non–changing fields like IP addresses, next header numbers, etc.) and non–constant (e.g., *Timestamp*, *IP Id*, etc.) functions. The non–*IR* packet types do not contain any chains, instead they append the extension related data, encapsulation data and also the *UDP checksum* value at the end.

The packet types of RoHC version 2 can be separated into two groups. The first group contains packets that are only sent during initialisation (*IR*) and during context repair (*IR* and *co_repair*). For ideal compression these packets are sent very rarely. The second group contains smaller packets that are most commonly used to update the decompressor

[2] With small CIds, the compression only has an id pool of 16 different numbers, whereas using large CIds, we have up to 16384 possible choices.

contexts. Their sizes range from 2 bytes (*pt_0_crc3*) to a maximum of about 12 bytes (*co_common*). As an example, Figure 3 contains some of the packet types employed and their respective sizes during the compression of a test stream.

The compressed packet choices for IPHC depend on whether the uncompressed stream contains TCP or UDP. For TCP, the compressor either uses *COMPRESSED_TCP* packets, which transmit delta values or *COMPRESSED_TCP_NODELTA*s, which contain most of the fields uncompressed and can be used to transmit changes that cannot be represented by delta encoding. For non–TCP headers only one packet type is defined, which can contain either an 8–bit CId or a 16–bit CId.

4. Methods and Metrics

Before evaluating the performance of RoHCv1, RoHCv2 and IPHC reference implementations provided by acticom GmbH[3], we introduce the main configuration and performance metrics employed in this paper. Without loss of generality, we focus our presentation on bandwidth savings and complexity in different scenarios utilising IPv4.

For the measurement testbed, we used two separate and different setups. For all UDP stream evaluations, we captured and stored the outgoing packets on the link layer. This approach enables consecutive evaluations with the same underlying data stream. We then applied artificial packet losses (if any) before decompression. Unlike the connection–based TCP, UDP does not define any feedback of its own, which facilitates this straight–forward configuration.

TCP, however, required a different approach. We employed two Raspberry Pi single–board computers that were equipped with WiFi modules. The devices were connected to each other via an ad–hoc connection and the Unix/Linux platform's TUN/TAP interfaces were utilised to create a tunnel to which we directed a normal TCP data stream from the Internet Layer. The benefit of this specific setup is that it enables a regular application to drive the compressed transmission. This way, any losses that were artificially induced would be handled directly by the Linux kernel after the redirection of the decompressed stream into the operating system's protocol stack. The Raspberry Pi platform was chosen due to broad availability, for its suitability as a platform for simulating nodes and because the platform had already shown the potential to support operations at high speeds. Specifically, this particular platform can be regarded as a general baseline for current mobile device capabilities.

Shifting our view to the performance evaluation metrics, an idealistic upper bound on the possible network bandwidth savings can be calculated by assuming that the compressed header size is zero. In this specific case, for each generated packet i the savings S_i are given by

$$S_i = 1 - \frac{P_i}{UH + P_i} = \frac{UH}{UH + P_i}, \tag{1}$$

where P_i denotes the payload data size of the i^{th} packet and UH denotes the size of the uncompressed protocol headers, i.e., the protocol encapsulation overhead. The average

[3] See `http://www.acticom.de`

savings for a sequence or session of N packets are in turn calculated as

$$\overline{S} = \frac{1}{N} \sum_{i=1}^{N} S_i, \tag{2}$$

describing the portion of the bandwidth that can be saved from a network provider's point of view. To illustrate this using an example, we consider the AMR codec with the smallest payload of 12 bytes. In an ideal scenario, one could assume that the compression of the protocol headers will result in zero bytes. Furthermore, assuming a full RTP/UDP/IPv4 or RTP/UDP/IPv6 encapsulation, the upper bound savings as given beforehand equal 77 % and 83 %, respectively.

For a more realistic scenario, we need to note that the compressed header size of an individual packet CH_i is greater than zero, i.e., $CH_i > 0$. In turn, we derive actual performance measurements similar to [39] as (i) the actual savings (or alternatively the gain) of the encapsulation overhead (headers) as

$$S_H(i) = \frac{UH - CH_i}{UH}, \tag{3}$$

(ii) the actual savings for individual packets as

$$S_P(i) = \frac{UH - CH_i}{UH + P_i}, \tag{4}$$

and (iii) the respective average savings according to Equation 2. In addition to the bandwidth savings, we evaluate the compression performance by means of the complexity of the compression/decompression methods employed in both RoHC versions and IPHC. We measure the complexity through CPU time–stamping, i.e., the time required to compress (or decompress) the i^{th} packet in the stream, as t_i and compare it to the previous time stamp obtained at time t_{i-1}. This complexity or utilised time could readily be mapped to mobile device power consumptions, see, e.g., [4], which, however, is out of the scope of this contribution. For our initial evaluation of RoHCv2's power consumption we refer to [44].

5. Compression of UDP Audio Streams

In this section, we first consider a channel without errors to illustrate the basic concepts for UDP transmissions. We additionally note that while in our evaluation we focus mostly on results regarding IPv4, the savings for IPv6 would be significantly higher.

5.1. Compression Savings for UDP Streams

We initially present results for real–world measurements using the two RoHC versions and IPHC over a wireless channel presumed to be error–free. To enable measurements under common real–world scenarios, we utilised the Asterisk VoIP server[4], which connected

[4] See `https://www.asterisk.org` for details.

a fixed desktop client and an Android smartphone, both using the ZoIPer VoIP client software[5]. This configuration employed the GSM 06.10 codec, which is the full–rate audio codec version that results in 33 bytes payload. We illustrate the resulting packet–indexed savings S_i in Figure 4.

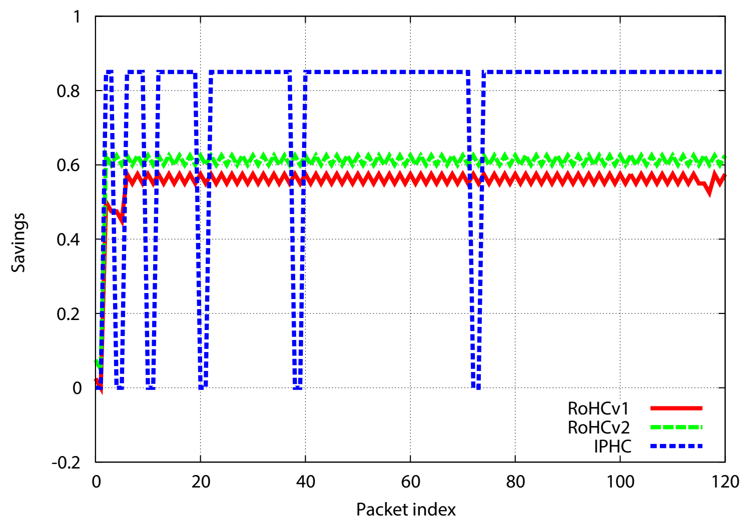

Fig. 4. Header savings $S_H(i)$ attained with the RoHCv1, RoHCv2 UDP profiles and the IPHC reference implementations for a VoIP call with GSM full rate codec using IPv4 over WLAN.

We observe that the compressed header savings typically range from 50 % to about 60 % for both RoHC versions. For IPHC, which exhibits occasional drops in savings due to sending uncompressed refreshes, we see around 85 % savings. We note that in this evaluation the best possible setup for each compression method was employed, i.e., while RoHC can rely on feedback to signal invalid contexts, IPHC lacks such functionality when compressing UDP. Therefore, IPHC must rely on periodic refreshes using an exponential function to tackle invalid contexts. Such mechanism is also defined for RoHC (employing the *optimistic approach*), which would also exhibit similar characteristics to IPHC. We also observe that RoHCv2 has a slightly better compression ratio than RoHCv1 (with about 5 % difference).

We attribute the seemingly worse performance of the RoHC versions when compared to IPHC (which is the earlier compression method) to the design of RoHC. IPHC only has to transmit the *Context Id* (1 byte), the *Generation Number* (1 byte) and the *UDP Checksum* (2 bytes) for the smaller packets, while in RoHC the compressor always has to send the *Master Sequence Number* (MSN, 2 bytes), various irregular IP fields (2–3 bytes), *CRC* (1 byte) and *UDP Checksum*, among other fields. Subsequently, and without loss

[5] See `http://www.zoiper.com` for details.

of generality, the smallest RoHC packet will be about at least 5 bytes larger than the corresponding IPHC packet.

We illustrate the header compression and total savings in Figure 5 for a similar audio–only scenario, but with IPv6 headers. We immediately observe that the compressed header savings for both RoHC versions are about 70 %, while IPHC shows 90 % header savings with the same periodical refreshes as before. We provide a numerical comparison between

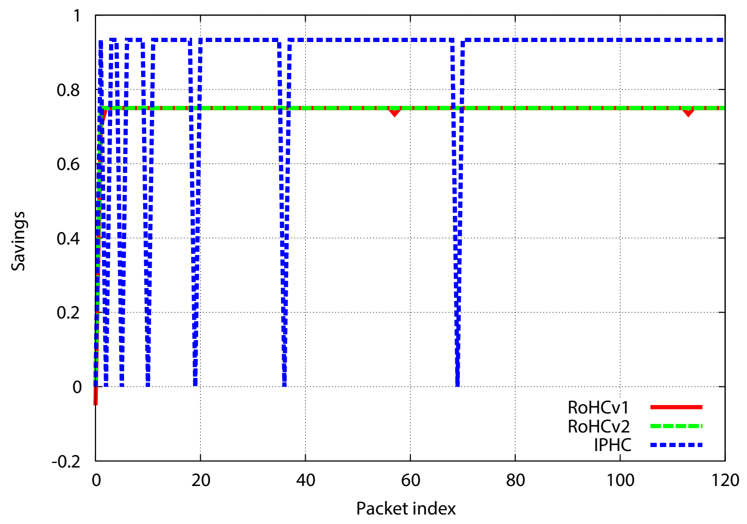

Fig. 5. Header savings $S_H(i)$ attained with the RoHCv1, RoHCv2 UDP profiles and the IPHC reference implementations for a VoIP call with GSM full rate codec using IPv6 over WLAN.

the two RoHC versions and IPHC, as well as the two IP versions in Table 2. We note that the average compressed header sizes attained with RoHCv1 and RoHCv2 are almost exactly the same for both IP versions. However, the average header size of IPHC is about 9 bytes smaller than than the ones observed for RoHC's compressed packets. Taking into account that IPHC relies on the link layer to signal packet types, it still outperforms RoHC by approximately 8 bytes. Comparing the packet savings in general, we note that IPHC with IPv6 is about 2 bytes better, because it can completely omit the transmission of the *IP Identification* field.

Overall, we conclude that both RoHC versions perform at about the same level. However, IPHC outperforms them easily by at least 25 % (not taking into account the mandatory periodic refreshes).

5.2. Compression Complexity for UDP Streams

Next, we evaluate the compression and decompression complexity by means of CPU–level timestamps, which can be mapped directly to CPU cycles utilised for (de)compression

Table 2. Average savings with the RoHCv1, RoHCv2 UDP profiles and IPHC for an audio streaming scenario over a reliable channel with IPv4 and IPv6.

IP version	**Compression**	**Uncompressed Header [byte]**	**Average Comp. Header [byte]**	$\overline{S_P}$	$\overline{S}$
v4	RoHCv1	28	15.03	13.23%	46.30%
	RoHCv2		15.02	13.24%	46.33%
	IPHC		6.29	33.00%	77.51%
v6	RoHCv1	48	15.05	14.54%	68.64%
	RoHCv2		15.03	14.56%	68.68%
	IPHC		4.48	51.17%	90.65%

and energy consumption in turn. We illustrate the compression timestamps for the VoIP call in Figure 6.

Comparing the different compression levels, we observe that RoHCv1 and IPHC exhibit excellent performance, as indicated by almost negligible times required for compression. On the other hand, we note that the compression of RoHCv2 is more CPU intense. We attribute this contrast to the more elaborate design of RoHCv2, which uses more complexity to attain better performance and robustness.

In Figure 7, we illustrate the corresponding decompression complexity. We initially observe that the timestamp levels are lower (by about 2 orders of magnitude) for all decompressions and both versions of RoHC. Comparing the RoHC versions to IPHC, we identify a significant difference between these approaches, as RoHC needs only an approximate quarter of the CPU time required by IPHC. However, we note that the scale is almost an order of magnitude smaller than for the compression. We furthermore note that the “spikes” in the timestamps of IPHC correspond to the periodic refreshes.

Overall, we additionally observe a fairly narrow range (except for minor outliers) in the (de)compression times, indicating a generally stable performance.

6. Compression of a TCP Acknowledgement Stream

In this section, we now shift the view to the compression performance achieved by employing the RoHC TCP profile and IPHC. We note that RoHC has only one defined TCP profile and the shift from RoHCv1 to RoHCv2 does not introduce differences for this scenario.

In Figure 8 we illustrate the compression ratios for a TCP acknowledgement stream of a digital radio station. We note that header compression generally exhibits very limited gains when the packet payloads are large, which normally is the case for TCP (segments are commonly full–framed from the link–layer’s point of view). However, acknowledge-

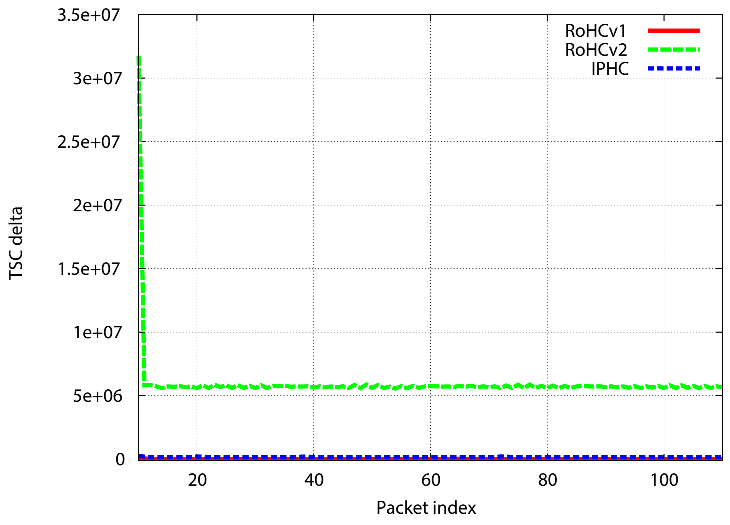

Fig. 6. Header compression complexity measured by CPU timestamps (TSC) attained with the RoHCv1, RoHCv2 UDP profiles and the IPHC reference implementations for a VoIP call with GSM full rate codec using IPv4 over WLAN.

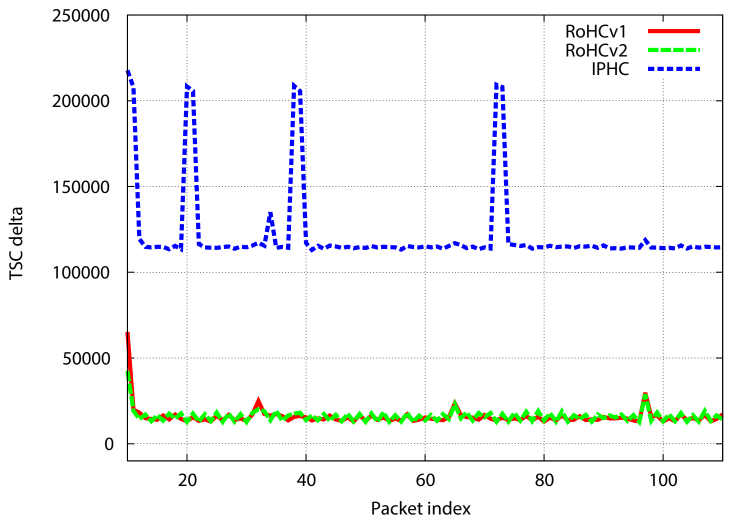

Fig. 7. Header decompression complexity measured by CPU timestamps (TSC) attained with the RoHCv1, RoHCv2 UDP profiles and the IPHC reference implementations for a VoIP call with GSM full rate codec using IPv4 over WLAN.

ment streams are ideally suitable candidates for compression, since they do not contain any significant amount of payload.

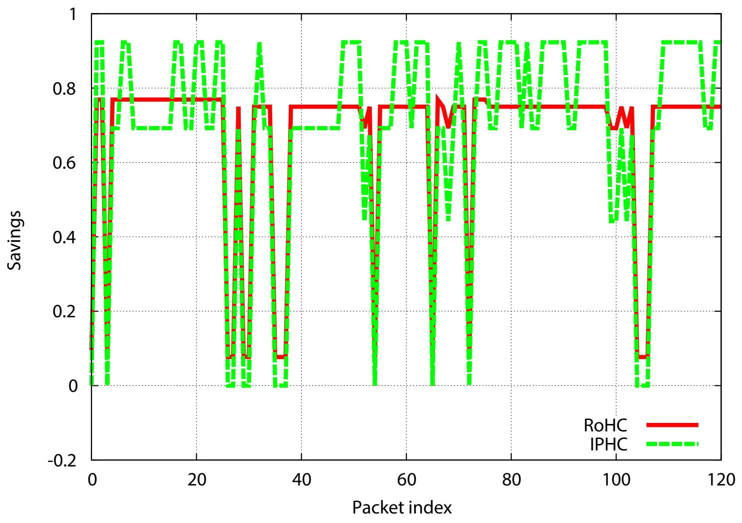

Fig. 8. Average header savings $S_H(i)$ attained with the RoHC TCP profile and the IPHC reference implementations for the acknowledgement stream of a digital radio transmission.

As we observe from the values presented in Figure 8, both RoHC and IPHC can result in savings of around 80 % for the header data. We note, however, that IPHC outperforms RoHC at its peak efficiency by about 10 % on average. However, IPHC also loses about 5–10 % in its periodical fallbacks to larger packet types. The downward spikes illustrated in Figure 8 are explained by the constantly changing TCP flags and various TCP option fields. Upon closer inspection, we note that RoHC is in general more capable at handling such “erratic” changes of the TCP header fields as it employs a version of table based list compression to maintain a history of previous option field values. Since IPHC lacks such functionality, it usually loses 5–20 % when compared to RoHC.

Figure 9 and Figure 10 illustrate the TCP complexity during compression and decompression for this scenario, respectively. We initially observe that the results are in the same magnitude as the previously discussed UDP measurements. However, the overall compression is faster, while decompression is slower than observed for the UDP stream. As outlined before for the compression efficiency, the increased number of TCP header fields and their exhibited variations over time are responsible for the increased time to compress the headers.

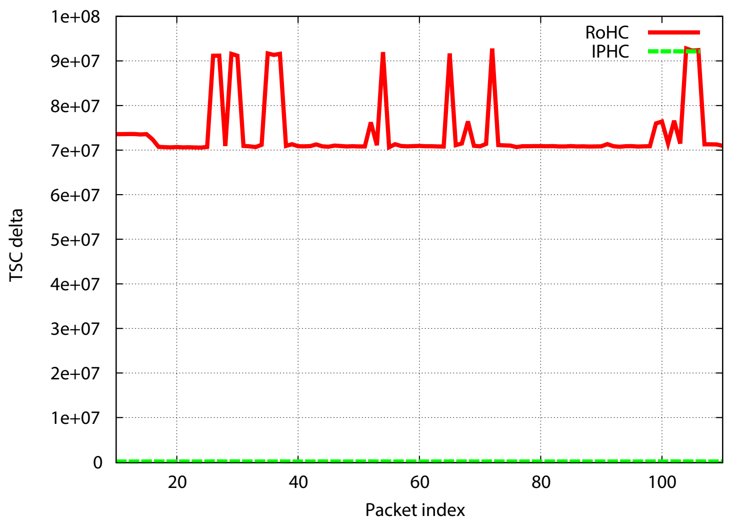

Fig. 9. Header compression complexity measured by CPU timestamps (TSC) attained with the RoHC TCP profile and the IPHC reference implementations for the acknowledgement stream of a digital radio transmission.

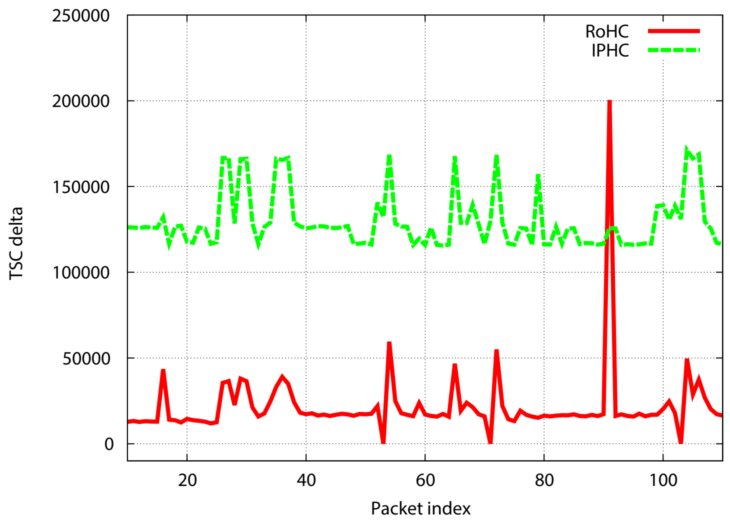

Fig. 10. Header decompression complexity measured by CPU timestamps (TSC) attained with the RoHC TCP profile and the IPHC reference implementations for the acknowledgement stream of a digital radio transmission.

7. Error–prone Channel Performance Using UDP

To evaluate the performance of the compression schemes under realistic conditions, in this section we consider a simulated wireless channel using an uncorrelated error model. Based on prior research relating to RTP profile measurements [42], we note that correlated errors only affect the compression savings if the loss rate of the channel is higher than 50 %. To maintain focus on the majority of use–cases, we look at uncorrelated error scenarios here.

Even though both RoHC versions and IPHC could perform more reliably than any other header compression schemes, e.g., those proposed by Van Jacobson [18] or Perkins and Mutka [32], there might be additional losses due to the lost decompressor state. In order to get a meaningful performance evaluation, burstiness is needed to get the decompressor out of sync. In earlier compression schemes, any error would make the decompressor lose synchronism with the compressor, but RoHC was designed to be more robust with respect to single failures.

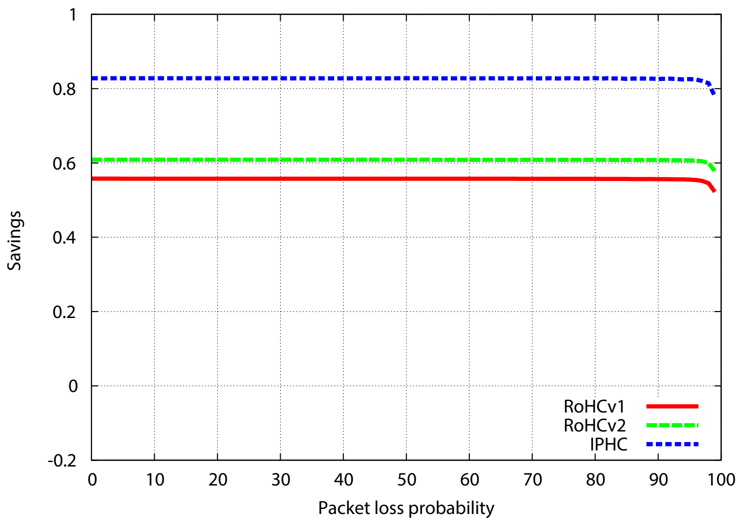

Fig. 11. Average header savings $S_H(i)$ attained with the RoHCv1, RoHv2 UDP profiles and the IPHC reference implementations for a UDP stream over an IPv4 link with different uncorrelated packet loss probabilities.

We consider the uncorrelated error–prone channel scenario and illustrate the header compression savings obtained in Figure 11. This approach is similar to the error–free measurements with RoHC and IPHC, illustrated in Figure 4. However, the major difference between the RoHC compression and IPHC schemes does not show up in the figure. Since the IPHC UDP profile does not have any feedback capability, the IPHC decompressor will on multiple occasions lose synchronisation with its compressor counterpart. This directly results in decompression failures if the next correctly received compressed packet is in a different generation. Therefore, packet losses may affect correctly received IPHC packets later on. In our test stream we did not observe any decompression failures with either RoHC versions.

We additionally note that the recommended IPHC compressor configurations optimise the periodic context refreshes very well. For loss–rates in the range of 0 % to 25 % approximately 0.001 % of the correctly received compressed packets are discarded because of invalid decompressor contexts. We furthermore note that longer streaks of decompression failures were observed as well, which are the result of out–of–sync decompressors.

8. Compression of Live TCP Streaming Sessions

In this section we shift the evaluation to the TCP compression savings utilising a different environment which is capable of compressing TCP streams online. Compared to the measurements discussed in the previous sections, we now show the compression efficiency when the compressed TCP stream is live. We perform this evaluation to demonstrate the compression of dynamic streams, i.e., a stream that was not previously captured on the network interface. This live stream, in turn, is also responding to various packet losses via kernel generated acknowledgements.

For the following tests we also turned on the handling of reordering in IPHC which forces the compressor to only use *COMPRESSED_TCP_NODELTA* packets that are generally larger than the *COMPRESSED_TCP* packets due to that they contain the whole uncompressed TCP header (except the port numbers). This is recommended by the corresponding RFC and is performed in order to avoid any decompression failures arising from retransmitted TCP packets by the kernel (in [18]: “11.2. Reordering in TCP packet streams”).

8.1. Compression of a World Wide Web Radio Transmission

We initially consider the compression savings obtained for a *World Wide Web* (WWW) radio transmission. We employed the VLC media player to re–stream the feed of a live WWW radio station. The audio payload was in the *mp3* format and was remuxed to a 32 kbit/sec bitrate with a sample rate of 8000 Hz.

In Figure 12 we illustrate the compressed stream that is received by the client application. We observe that in contrast to the results observed for the static environment presented in Section 6, the IPHC compression savings here are significantly smaller than those observed with RoHC. We attribute this change partially to the constraint of transmitting only *COMPRESSED_TCP_NODELTA* packets, which are needed for the retransmission of lost packets in order to tackle reordering. In this case, RoHC achieves more than 80 % savings, while IPHC only achieves about 60 %.

We illustrate the acknowledgement stream sent by the client’s operating system back to the server in Figure 13. We initially observe the same 20 % difference between the two general compression approaches. However, the compression savings obtained here are about 5–10 % smaller than those observed for the server stream. This behaviour is most likely rooted in the larger TCP options list, which is generally present in TCP acknowledgements (e.g., *SACK* fields).

We additionally observe the initial ramp–up period at the start of the streaming in Figure 13. This is the time period where the TCP connection is established between the

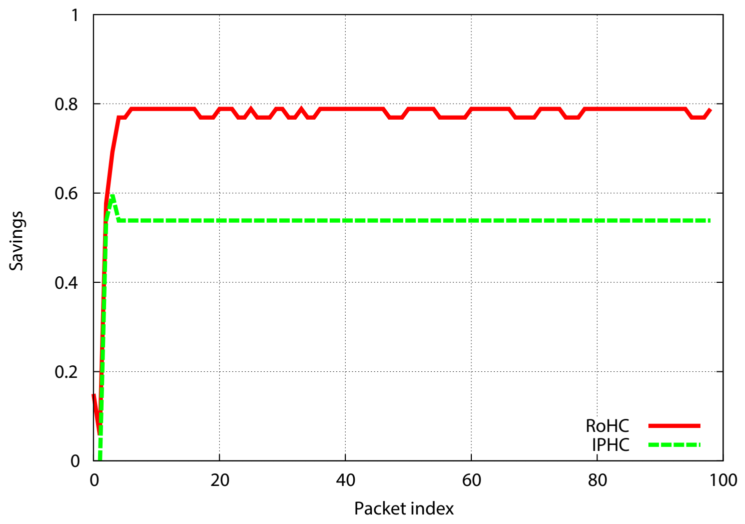

Fig. 12. Header savings $S_H(i)$ attained with the RoHC TCP profile and the IPHC reference implementations for a radio stream over an IPv4 link containing the audio payload (downstream).

server and the client; it, in turn, corresponds to the three–way handshake used by the TLS protocol.

8.2. Error–Prone Channel Performance Using TCP

We now look at the evaluation of the compression savings when artificial packet losses are introduced. For the simulation of such losses, we employ an uncorrelated randomised algorithm which discards compressed packets before decompression. For this scenario, the corresponding feedback mechanisms are enabled for both IPHC and RoHC. The server application that is in charge of maintaining TCP creates a connection and sends predetermined data to the client (downstream). The client, in this case, only acknowledges the received packets (upstream).

For these measurements, we compressed 100 packets 10 times and calculated the mean savings with 95 % confidence intervals for each measured loss–rate. We illustrate the resulting savings for the different compression mechanisms in Figure 14. We observe that the compression savings are inverse proportional to the loss–rate in the case of the server stream. In the measurement time interval, we additionally observe that at 10 % loss–rates, the compression savings are approximately 10 % lower than in a scenario without losses. This observation is applicable for both, RoHC and IPHC. We additionally observe that the IPHC packets are again about 20 % larger than the packets produced by RoHC.

In Figure 15 we illustrate the acknowledgement stream as sent by the client (upstream) to the server. We observe that the decrease in savings are insignificant in the measured time intervals when compared to the downstream compression.

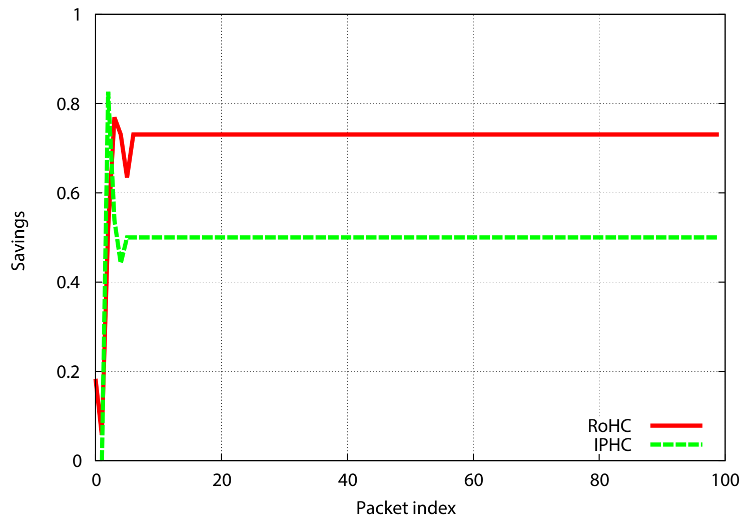

Fig. 13. Header savings $S_H(i)$ attained with the RoHC TCP profile and the IPHC reference implementations for a radio stream over an IPv4 link containing the TCP acknowledgements (upstream).

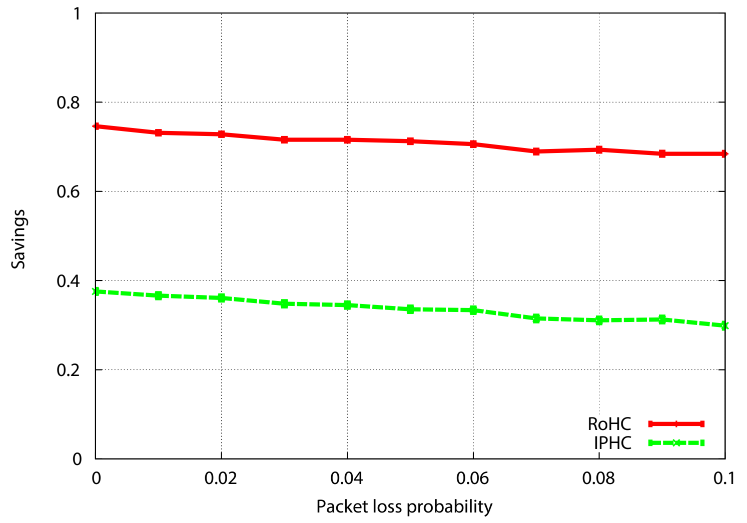

Fig. 14. Average header savings $S_H(i)$ attained with the RoHC TCP profile and the IPHC reference implementations for a radio stream over an IPv4 link containing the audio payload (downstream) with different uncorrelated packet loss probabilities.

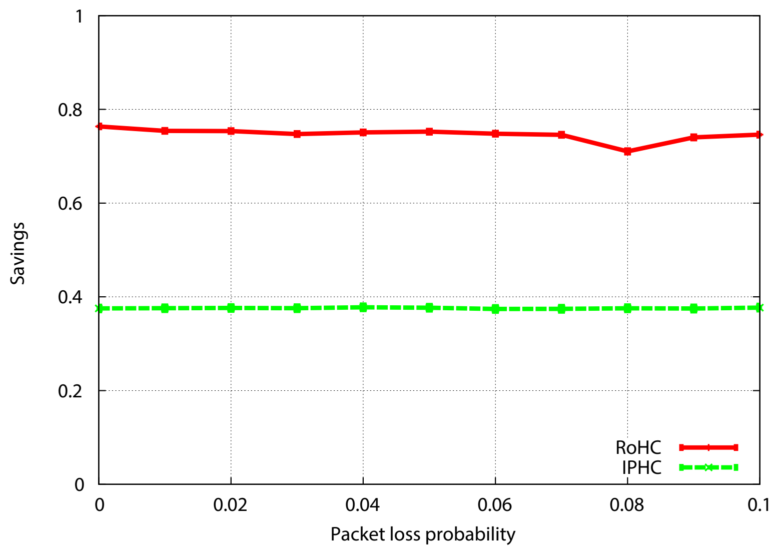

Fig. 15. Average header savings $S_H(i)$ attained with the RoHC TCP profile and the IPHC reference implementations for a radio stream over an IPv4 link containing the TCP acknowledgements (upstream) with different uncorrelated packet loss probabilities.

In both cases, we observe that the confidence intervals are very tight, which corresponds to the observation that, on average, the compressed packet sizes are very close to the mean size. Looking at the raw data, this usually refers to about 1–2 bytes difference between compressed packets.

9. Conclusion

Throughout this article, we compared and contrasted several real–world performance measurements for Robust Header Compression (RoHC) version 1, RoHC version 2 and IP Header Compression (IPHC). Additionally, we presented measurements of a live compression performed over a wireless tunnel. Overall, we conclude that IPHC is capable of achieving approximately 10 % to 20 % percent higher UDP/IP compression than either RoHC version. However, IPHC potentially aggravates errors locally due to it lacking an appropriate feedback mechanism. With the correct compressor configuration these could be minimised sufficiently. Compressing streams with UDP/IP compression has the additional benefit of having a stable and persistent compressed header size which is at least the fourth of the original uncompressed size. Between the two RoHC versions, the second edition consistently shows better compression gains by 5–10 %.

RoHC and IPHC TCP compressions are quite similar to each other regarding their compression gain when one is not concerned with reordering. Albeit the IPHC benefits from a simpler compressed packet structure, which results in faster compression, it does not compress TCP fields when reordering is expected. This subsequently results in decreased header compression savings, which are roughly half of what Robust Header

Compression can achieve. In scenarios where reordering is of no concern, IPHC will commonly perform equally or even slightly better when compared to RoHC. Our results also indicate that while RoHCv2 uses significantly more complexity during compression than either IPHC (both for UDP and TCP) or RoHC (in case of UDP), it is somewhat faster when decompressing.

Since Robust Header Compression is primarily designed for the compression of live audio transmissions on endpoint devices, it performs best if certain fields, such as the identification numbers and timestamps, stay constant or change at a predetermined rate. Moreover, the compressor expects the uncompressed stream to be loss–free as well. Albeit RoHC achieves around 80 %–90 % gain overall, neither of the compression designs were optimised for general use. Our current research efforts focus on finding the optimal configuration of RoHC based on the channel and uncompressed stream characteristics online. We employ machine learning approaches for the prediction of header compression utilities in [41], which will enable the compression to dynamically adapt to varying channel conditions. This will ensure that the compression always performs at best efficiency throughout the transmission. Furthermore, we are working on various ways to combine RoHCv2 and network coding to maximise throughput while reducing the potential latency in real–time mesh networks.

Acknowledgments. The authors thank acticom GmbH, especially Gerrit Schulte, for the support and software reference implementations of RoHCv1, RoHCv2 and IPHC as well as their help in conducting the experiments. This work was supported by the János Bolyai Research Fellowship of the Hungarian Academy of Sciences.

References

1. Arango, J., Pink, S., Ali, S., Hampel, D.: Header compression for ad-hoc networks. In: Military Communications Conference, 2005. MILCOM 2005. IEEE. pp. 3080–3086 Vol. 5 (Oct 2005)
2. Benhassine, N., Thierry, E., Bonnin, J.M.: Efficient header compression implementation for ip-based its communications. In: ITS Telecommunications (ITST), 2012 12th International Conference on. pp. 780–784 (Nov 2012)
3. Bormann, C., Burmeister, C., Degermark, M., Fukushima, H., Hannu, H., Jonsson, L.E., Hakenberg, R., Koren, T., Le, K., Liu, Z., Martensson, A., Miyazaki, A., Svanbro, K., Wiebke, T., Yoshimura, T., Zheng, H.: RObust Header Compression (ROHC): Framework and four profiles: RTP, UDP, ESP, and uncompressed (July 2001), request for Comments 3095
4. C. Yoon, D. Kim, W.J.C.K., Cha, H.: Appscope: application energy metering framework for android smartphones using kernel activity monitoring. In: Proceedigs of 2012 USENIX conference on Annual Technical Conference. pp. 36–36. Berkeley, CA, USA (2012)
5. Calveras, A., Arnau, M., Paradells, J.: An improvement of tcp/ip header compression algorithm for wireless links. Third World Multiconference on Systemics, Cybernetics and Informatics (SCI'99) and the Fifth International Conference on Information Systems Analysis and Synthesis (ISAS'99), IEEE, Orlando, USA vol. 4, pp. 39–46 (July/August 1999)
6. Calveras, A., Paradells, J.: Tcp/ip over wireless links: Performance evaluation. 48th An-nual Vehicular Technology Conference VTC '98 IEEE, Ottawa (Ontario), Canada vol. 3, pp. 1755–1759 (May 1998)
7. Casner, S., Jacobson, V.: Compressing ip/udp/rtp headers for low-speed serial links. Request for Comments 2508 (1999)

8. Chen, W.T., Chuang, D.W., H.-C.Hsiao: Enhancing crtp by retransmission for wireless networks. Proceedings of the Tenth International Conference on Computer Communications and Networks pp. pp. 426–431 (2001)
9. Chen, X., Guo, F., Dang, P., Wu, L.: A survey of rohc header compression schemes. In: Computer Science and Network Technology (ICCSNT), 2012 2nd International Conference on. pp. 331–335 (Dec 2012)
10. Cheng, B.N., Moore, S.: Securing robust header compression (rohc). In: Military Communications Conference, MILCOM 2013 - 2013 IEEE. pp. 1383–1390 (Nov 2013)
11. Cheng, B.N., Wheeler, J., Hung, B.: Internet protocol header compression technology and its applicability on the tactical edge. Communications Magazine, IEEE 51(10), 58–65 (October 2013)
12. Cheng, B.N., Wheeler, J., Hung, B., Moore, S., Sukumar, P.: A comparison of ip header compression schemes in manets. In: Performance Computing and Communications Conference (IPCCC), 2013 IEEE 32nd International. pp. 1–9 (Dec 2013)
13. Cheng, B.N., Zuena, J., Wheeler, J., Moore, S., Hung, B.: Manet ip header compression. In: Military Communications Conference, MILCOM 2013 - 2013 IEEE. pp. 494–503 (Nov 2013)
14. Fitzek, F.H., Rein, S., Seeling, P., Reisslein, M.: Robust header compression (rohc) performance for multimedia transmission over 3g/4g wireless networks. Wireless Personal Communications 32(1), 23–41 (2005)
15. Fortuna, P., Ricardo, M.: Header compressed voip in ieee 802.11. IEEE Wireless Communications 16(3), 69–75 (June 2009)
16. Hermenier, R., Kissling, C.: Optimization of robust header compression for aeronautical communication. In: Integrated Communications, Navigation and Surveillance Conference (ICNS), 2013. pp. 1–11 (April 2013)
17. Hermenier, R., Rossetto, F., Berioli, M.: On the behavior of robust header compression u-mode in channels with memory. IEEE Transactions on Wireless Communications 12(8), 3722–3732 (August 2013)
18. Jacobson, V.: Compressing tcp/ip headers for low-speed serial links. Request for Comments 1144 (1990)
19. Jin, H., Hsu, R., Wang, J.: Performance comparison of header compression schemes for rtp/udp/ip packets. In: Wireless Communications and Networking Conference, 2004. WCNC. 2004 IEEE. vol. 3, pp. 1691–1696 Vol.3 (March 2004)
20. Jivorasetkul, S., Shimamura, M., Iida, K.: Better network latency with end-to-end header compression in sdn architecture. In: Communications, Computers and Signal Processing (PACRIM), 2013 IEEE Pacific Rim Conference on. pp. 183–188 (Aug 2013)
21. Jonsson, L.E., Pelletier, G.: RObust Header Compression (ROHC): A Compression Profile for IP (June 2004), request for Comments 3843
22. Jung, S., Hong, S.: Network/hardware cross-layer evaluation for rohc and packet aggregation on wireless mesh networks. Wirel. Netw. 15(8), 1086–1101 (Nov 2009), `http://dx.doi.org/10.1007/s11276-008-0104-7`
23. Kidston, D.: Ip header compression and packet aggregation in mobile tactical networks. In: MILCOM 2009 - 2009 IEEE Military Communications Conference. pp. 1–7 (Oct 2009)
24. Ludovici, A., Calveras, A., Catalan, M., Gómez, C., Paradells, J.: Implementation and evaluation of the enhanced header compression (iphc) for 6lowpan. In: Proceedings of the 15th Open European Summer School and IFIP TC6.6 Workshop on The Internet of the Future. pp. 168–177. EUNICE '09, Springer-Verlag, Berlin, Heidelberg (2009), `http://dx.doi.org/10.1007/978-3-642-03700-9_18`
25. Maeder, A., Felber, A.: Performance evaluation of rohc reliable and optimistic mode for voice over lte. In: Vehicular Technology Conference (VTC Spring), 2013 IEEE 77th. pp. 1–5 (June 2013)

26. Naidu, D., Tapadiya, R.: Implementation of header compression in 3gpp lte. In: Information Technology: New Generations, 2009. ITNG '09. Sixth International Conference on. pp. 570–574 (April 2009)
27. Nascimento, A.G., Mota, E., Queiroz, S., Galvao, L., Nascimento, E.: Towards an efficient header compression scheme to improve voip over wireless mesh networks. In: Computers and Communications, 2009. ISCC 2009. IEEE Symposium on. pp. 170–175 (July 2009)
28. Pelletier, G.: RObust Header Compression (ROHC): Profiles for User Datagram Protocol (UDP) Lite (April 2005), request for Comments 4019
29. Pelletier, G., Sandlund, K.: RObust Header Compression Version 2 (ROHCv2): Profiles for RTP, UDP, IP, ESP and UDP-Lite (April 2008), request for Comments 5225
30. Pelletier, G., Sandlund, K., Jonsson, L.E., West, M.: RObust Header Compression (ROHC): Profiles for User Datagram Protocol (UDP) Lite (July 2007), request for Comments 4996
31. Perkins, S.J., Mutka, M.W.: Dependency removal for transport protocol header compression over noisy channels. International Conference on Communications (ICC), Montreal, Canada vol. 2, pp. 1025–1029 (June 1977)
32. Perkins, S.J., Mutka, M.W.: Dependency removal for transport protocol header compression over noisy channels. In: Proceedigs of IEEE International Conference on Communications (ICC). pp. 1025–1029. Montreal, Canada (1997)
33. Pinola, J., Piri, E., Pentikousis, K.: On the performance gains of voip aggregation and rohc over a wirelessman-ofdma air interface. In: Global Telecommunications Conference, 2009. GLOBECOM 2009. IEEE. pp. 1–6 (Nov 2009)
34. Quintana-Ramirez, I., Saldana, J., Ruiz-Mas, J., Sequeira, L., Fernandez-Navajas, J., Casadesus, L.: Optimization of p2p-tv traffic by means of header compression and multiplexing. In: Software, Telecommunications and Computer Networks (SoftCOM), 2013 21st International Conference on. pp. 1–5 (Sept 2013)
35. Rein, S., Reisslein, M., Fitzek, F.H.P.: Voice quality evaluation for wireless transmission with rohc. Tech. rep., in International Conference on Internet and Multimedia Systems and Applications (2003)
36. Ribičre, M., Charlton, P.: Cisco visual networking index: Global mobile data traffic forecast update. Cisco, Inc. (2014–2019), [Online]. Available: http://www.cisco.com/c/en/us/solutions/service-provider/visual-networking-index-vni/index.html (current Feb. 2015.)
37. Rossi, M., Giovanardi, A., Zorzi, M., , Mazzini, G.: Tcp/ip header compression: Proposal and performance investigation on a wcdma air interface. 12th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, IEEE vol. 1, pp. A–78 – A–82 (September 2001)
38. Rossi, M., Giovanardi, A., Zorzi, M., Mazzini, G.: Improved header compression for tcp/ip over wireless links. Electronics Letters vol. 36(no. 23), pp. 1958–1960 (November 2000)
39. Seeling, P., Reisslein, M., Fitzek, F., Hendrata, S.: Video quality evaluation for wireless transmission with robust header compression. In: Information, Communications and Signal Processing, 2003 and Fourth Pacific Rim Conference on Multimedia. Proceedings of the 2003 Joint Conference of the Fourth International Conference on. vol. 3, pp. 1346–1350 vol.3 (Dec 2003)
40. Svanbro, K., Hannu, H., Jonsson, L.E., Degermark, M.: Wireless real–time ip services enabled by header compression. Proceedings of the IEEE Vehicular Technology Conference (VTC), Tokyo, Japan vol. 2, pp. 1150–1154 (2000)
41. Tömösközi, M., Seeling, P., Ekler, P., Fitzek, F.H.P.: Regression model building and efficiency prediction of rohcv2 compressor implementations for voip. In: 2016 IEEE Global Communications Conference (GLOBECOM). pp. 1–6 (Dec 2016)
42. Tömösközi, M., Seeling, P., Fitzek, F.H.: Performance evaluation and comparison of robust header compression (ROHC) rohcv1 and rohcv2 for multimedia delivery. In: Workshops Pro-

ceedings of the Global Communications Conference, GLOBECOM. pp. 1346–1350. Atlanta, GA, USA (2013)

43. Tömösközi, M., Seeling, P., Ekler, P., Fitzek, F.H.: Applying robust header compression version 2 for udp and rtp broadcasting with field constraints. In: IEEE 85rd Vehicular Technology Conference, VTC Spring 2016. Sydney, Australia (Jun 2016)
44. Tömösközi, M., Seeling, P., Ekler, P., Fitzek, F.H.: Robust header compression version 2 power consumption on android devices via tunnelling. In: ICC2017: WS05-International Workshop on Application of green techniques to emerging communication and computing paradigms (GCC)" (ICC2017-WS05). Paris, France (May 2017)
45. Tordjman, T., Lücke, O.: Evaluation of robust header compression for aeronautical operational data. In: 2012 6th Advanced Satellite Multimedia Systems Conference (ASMS) and 12th Signal Processing for Space Communications Workshop (SPSC). pp. 308–315 (Sept 2012)
46. Woo, H., Kim, J., Lee, M., Kwon, J.: Performance analysis of robust header compression over mobile wimax. In: Advanced Communication Technology, 2008. ICACT 2008. 10th International Conference on. vol. 3, pp. 1742–1746 (Feb 2008)

Máté Tömösközi is a Junior Researcher at the Technische Universität Dresden (Dresden, Germany), as well as co–Ph.D. at the Budapest University of Technology and Economics (Budapest, Hungary) and the Aalborg University (Aalborg, Denmark). He received his master's degree in Computer Engineering from the Budapest University of Technology and Economics in 2013 and worked as a software engineer for acticom GmbH (Berlin, Germany) between 2012–2015 where he developed and maintained various header compression implementations. He is currently researching header compression and network coding in the context of next generation networks (5G) with an emphasis on latency in real–time mesh networks.

Patrick Seeling is an Associate Professor in the Department of Computer Science at Central Michigan University (Mount Pleasant, Michigan, USA). He received his Dipl.-Ing. Degree in Industrial Engineering and Management from the Technical University of Berlin (Berlin, Germany) in 2002 and his Ph.D. in Electrical Engineering from Arizona State University (Tempe, Arizona, USA) in 2005. He currently leads the Distributed Internetworked Systems and Content (DISC) lab at Central Michigan University, reflecting his research interests comprising user experiences in mixed realities, networking (with a focus on multimedia and energy optimisations), distributed and mobile systems, and computer–mediated education. Patrick Seeling is a Senior Member of the Association for Computing Machinery (ACM) and the Institute of Electrical and Electronics Engineers (IEEE).

Péter Ekler is a Senior Lecturer at the Budapest University of Technology and Economics, Department of Automation and Applied Informatics. He received his Ph.D. degree at BME in 2011. He has been working with mobile P2P and social networks for six years. He is the creator of the first BitTorrent client for mainstream mobile phones based on the Java ME platform. He is co–author of several mobile related scientific papers and book chapters. His field of research covers mobile–based social networks, P2P solutions, data analysis and power law distributions in large networks. He has participated in several data warehouse and business intelligence related projects. He teaches mobile software development for several mobile platforms.

Frank H.P. Fitzek is a Professor and chair of the Deutsche Telekom chair of communication networks group at Technische Universität Dresden coordinating the 5G Lab Germany. He received his diploma (Dipl.-Ing.) degree in electrical engineering from the University of Technology - Rheinisch–Westfälische Technische Hochschule (RWTH) - Aachen, Germany, in 1997 and his Ph.D. (Dr.-Ing.) in Electrical Engineering from the Technical University Berlin, Germany in 2002 and became Adjunct Professor at the University of Ferrara, Italy in the same year. In 2003 he joined Aalborg University as Associate Professor and later became Professor. He co–founded several start–up companies starting with acticom GmbH in Berlin in 1999. He has visited various research institutes including Massachusetts Institute of Technology (MIT), VTT, and Arizona State University. In 2005 he won the YRP award for the work on MIMO MDC and received the Young Elite Researcher Award of Denmark. He was selected to receive the NOKIA Champion Award several times in a row from 2007 to 2011. In 2008 he was awarded the Nokia Achievement Award for his work on cooperative networks. In 2011 he received the SAPERE AUDE research grant from the Danish government and in 2012 he received the Vodafone Innovation price. In 2015 he was awarded the honorary degree ”Doctor Honoris Causa” from Budapest University of Technology and Economics (BUTE). His current research interests are in the areas of wireless and mobile 5G communication networks, mobile phone programming, network coding, cross layer as well as energy efficient protocol design and cooperative networking.

Received: November 1, 2015; Accepted: February 27, 2017.