Acta Cybernetica **23** (2017) 737–756.

Prediction of RoHCv1 and RoHCv2 Compressor Utilities for VoIP

Máté Tömösközi[ac], Patrick Seeling[b], Péter Ekler[a], and Frank H.P. Fitzek[c]

Abstract

Modern cellular networks utilising the long–term evolution (LTE) and the coming 5G set of standards face an ever–increasing demand for low–latency mobile data from connected devices. Header compression is employed to minimise the overhead for IP–based cellular network traffic, thereby decreasing the overall bandwidth usage and, subsequently, transmission delays. Here, we employ machine learning approaches for the prediction of Robust Header Compression version 1's and version 2's compression utility for VoIP transmissions, which allows the compression to dynamically adapt to varying channel conditions.

We evaluate various regression models employing r^2 and mean square error scores next to complexity (number of coefficients) based on an RTP specific training data set and separately captured live VoIP audio calls. We find that the proposed weighted Ridge regression model explains about at least 50 % of the observed results and the accuracy score may be as high as 94 % for some of the VoIP transmissions.

Keywords: robust header compression, mobile multimedia, cellular networks, bandwidth savings, linear regression, machine learning

1 Introduction

Nowadays IP–based communication is prominent in various industry applications worldwide and it is even expected to increase as indicated, e.g. in [20]. With the wide distribution of LTE enabled smart–phones and Internet capable devices (IoT), as well as the adoption the coming 5G standard, user and machine alike are constantly connected to service providers, regardless whether of they are motionless

[a]Department of Automation and Applied Informatics, Budapest University of Technology and Economics, Hungary, E-mail: [mate.tomoskozi, ekler.peter]@aut.bme.hu

[b]Department of Computer Science, Central Michigan University, Mount Pleasant, MI 48859, USA, E-mail: pseeling@ieee.org

[c]Communication Networks Group, Technische Universität Dresden, Germany, E-mail: [mate.tomoskozi, frank.fitzek]@tu-dresden.de

DOI: 10.14232/actacyb.23.2.2017.17

or in motion. For real–time IP–based wireless connections, packets have to be sent in such a way that the users and applications experience only a minimal degradation in service quality.

During wireless communication the limited bandwidth and the relatively higher rate of errors require special attention. Since there is a large inequality between packet payloads and protocol headers during transmissions, there exists a common packetisation overhead for each packet. As an example, for a normal audio–only session the operators can save more than half of the data transmission costs by compressing various fields in the protocol headers above the link layer. This is possible because most of the fields in an IP traffic contain values that are constant or increase in a predetermined way during a transmission.

The reduction of this packetisation overhead sent over cellular and other wireless networks is typically realised through header compression mechanisms. Header compression approaches reduce the protocol encapsulation overhead between, say, wireless base stations and mobile clients. The compression approach is commonly based on a compressor/decompressor concept operating between the data link layer and the network layer on a sender/receiver pair's protocol stack implementations.

The first IP header compression scheme was the *Compressed Transport Control Protocol* (CTCP or VJHC). It was proposed by Van Jacobson [11] and it only handles the TCP protocol. CTCP combines TCP and IP headers together for better results and lower complexity. The compression algorithm itself employs *delta coding*. The advantage of this approach is the high compression ratio. Unfortunately, it is very susceptible to bit errors, which results in the dropping of many of the following packets by the receiver and there is no built–in error detection; instead it relies on the protection schemes of the lower and higher level protocols.

An improvement on this was introduced by *Perkins* in [19]. The delta coding for the neighbouring packets is replaced by a reference frame, much like modern video compressions. In this case, the first packet of a frame is sent as–is and the following packets use the delta coding to refer to the first one. This results in better tolerance to errors compared to CTCP, but it produces a lower compression gain. An improvement for the above approach by *Calveras* ([3] and [4]) proposes a dynamic frame length scheme as a function of the channel state. Both of these approaches suffer from desynchronisation when the first (uncompressed) packet is lost, which results in the corruption of all of the packets in the same frame. A suggested improvement is available from *Rossi* ([22] and [21]).

As the next step, the compression of RTP was addressed with the development of the *Compressed Real Time Protocol* (CRTP) [5]. *RObust Checksum-based COmpression* (ROCCO) [24] is a refinement of CRTP, which improves the header compression performance for highly error–prone links and long round–trip times. In a similarly way, *Enhanced Compressed RTP* (ECRTP) [6] is a refinement of CRTP.

Robust Header Compression (RoHC) builds on these predecessors and version 1 of RoHC, which was introduced in [2], and was modelled on the concept of extensibility with various profiles that were added later (see IP [12], UDP-Lite [16] and TCP profiles [18]). However, version 2 of Robust Header Compression, defined

in [17], opts for simplicity in design over extensibility.

For a brief overview of these header compression schemes see [7] or [25]. RoHCv1 was incorporated into 3GPP–UMTS and WiMAX networks with significant industry support to date and recently revised as RoHCv2 in RFC 5225 [17]. The new version increases the use–cases of RoHCv1 and it includes the enhancements proposed in–between in RFCs 3843, 4019, 4224, 4995, or 4997, without rendering any parts obsolete. The main goals of the augmented RoHCv2 are increased simplicity and robustness under similar network conditions, while maintaining or increasing performance. For a short review of Robust Header Compression's technical details, see [25].

Although both versions of RoHC achieve around 80–90 % in gain (see [28] and [26]), in spite of this neither of these compression designs accounts for the varying channel conditions that may be encountered in wireless setups (e.g., as a result of weak signals or interference). Specifically, the RFC for RoHCv2 in [17] states that "A compressor always ... repeats the same type of update until it is fairly confident that the decompressor has successfully received the information." and "The number of repetitions that is needed to obtain this confidence is normally related to the packet loss and out–of–order delivery characteristics of the link where header compression is used; it is thus not defined in this document [RFC 5525]." The impact of these repetitions is closely related to the protocol field dynamics of compressed IP–streams and it will differ from application to application.

Our present research efforts seek to address this issue by enabling current compressor implementations to configure themselves online, thereby making the compression adaptable to changing channel conditions and various network stream characteristics, which will in turn minimise compression overhead and it will pave the way for the adaptation of header compression in IoT scenarios.

Previous efforts regarding optimal configuration of the compression have only targeted the first version of RoHC. For employment with UMTS, the authors of [14] and [31] have studied and recommended optimal configurations for each compression mode to balance performance and robustness, while [15] relies on QoS information derived from the RRC layer of the radio link to dynamically configure the compression. The authors of [13] have proposed an adaptive optimistic approach, where the compressor switches to a high number of context refreshes when a negative feedback is received and uses a timer to return to the initial refresh count if no further errors occur. Again for RoHCv1, the authors of [10] use the idea of adjusting the sliding window width and the (re)initialisation timeout setting for specific (aeronautical) scenarios.

Here our current study, however, focuses on the optimal configuration of both RoHCv1 and RoHCv2 in an implementation independent and RFC compliant way which also exploits various header field dynamics to further the application of header compression for non–audio transmission use–cases as well. In [27], we investigated how the compressions react to different fluctuations in the headers and reviewed the compressor configurations which maximise utility. We showed that the appropriate choice of compressor repetition configuration increases the overall utility under dynamic channel conditions and finding the optimal configuration

might produce significant benefits. Our preliminary analysis of compression utilities and their prediction with linear regression in [29] resulted in limited, but viable prediction accuracy. Building on these we proposed a profiling scheme for RoHCv2 in [30] that provides a higher overall accuracy and we evaluated the attainable gains by employing a dynamic configuration using linear regression with polynomial basis functions.

In this study, we survey and extend our method to include version 1 of Robust Header Compression and compare the results provided by various regression methods including Support Vector Regression. These results may be employed in the RoHC compression process to dynamically adapt to changing channel conditions. Following our preliminaries, we introduce key utility concepts in Section 2. A description of the experimental setup is shown in Section 3, then in Section 4 we present our results. Finally in Section 5 we provide a brief summary and make some suggestions for future study.

2 Methods and Metrics

Below we evaluate the gains possible with dynamic configuration based on various measurements with the RoHCv1 and RoHCv2 reference implementations provided by acticom GmbH[1] for RTP traffic. We utilise systematically generated and real–life captured streams to allow consecutive evaluations with the same underlying data streams. These streams were replayed and RoHC–compressed before artificial packet losses were applied (if any). The resulting packet stream was sent to the RoHC decompressor. This is adequate since UDP does not define any feedback of its own. For the creation of our data sets we repeated this process with multiple streams and different configuration–option permutations.

Let P_i denote the payload data size of the i^{th} packet, UH_i denote the size of the i^{th} uncompressed protocol headers, and CH_i denote its compressed header size. Here we shall define actual performance measures, similar to those described in [23], as (i) the actual savings (or alternatively the gain) of the encapsulation overhead (headers), namely

$$SH_i = \frac{UH_i - CH_i}{UH_i}, \tag{1}$$

(ii) the actual savings for individual packets (with payload) as

$$SP_i = \frac{UH_i - CH_i}{UH_i + P_i}, \tag{2}$$

and (iii) the respective average savings for a sequence or session of N packets as

$$\overline{S} = \frac{1}{N} \sum_{i=1}^{N} S_i, \tag{3}$$

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

where the latter describes the bandwidth savings from a network provider point of view.

In order to accurately predict the usefulness of the compression, we need to take into account parameters that potentially decrease the overall compression gain. In this study, we define a utility function as

$$u_i(\mathbf{x}) = \{\mathbf{x} \in \mathbb{R}^k : u_i(x) \in [-1.0, 1.0] \subset \mathbb{R} \text{ and } k, i \in \mathbb{Z}\},$$

where $\mathbf{x}$ is a vector representing the k observed (independent) variable values for a given packet i. A u_i value of 1.0 is interpreted as the optimal utility and a -1.0 value as the most disadvantageous.

In our view, conditions that counterbalance optimal utility are decompressor feedback and decompression failure. During compression, feedback is produced by the decompressor and sent back to the compressor entity either on a dedicated channel or piggybacked onto a compressed packet (in the case of a peer–to–peer setup). Since compression feedback is generated by the compression layer and not as part of the original stream, each feedback byte decreases the overall gain. If there are decompression failures (commonly due to decompressor desynchronisation caused by extensive packet losses on the channel), each compressed packet failing decompression is sent completely in vain (the RoHC RFC forbids the interpretation of such packets). In this case, each byte of the compressed packet decreases the compression gain as well, since the transmission of such packets wastes bandwidth.

We will consider the following approach for the generation of the utility function. Let SH_i denote the savings of the i^{th} packet as defined in Equation 1, and FB_i denote the ratio between the size of the feedback generated after the decompression of the same packet and the length of the corresponding uncompressed header. The the utility of the i^{th} packet is

$$U_i = \sum_{i=0}^{N} (\phi(i) \cdot a \cdot SH_i - b \cdot FB_i), \tag{4}$$

where the coefficients a and b allow fine–tuning of the cost of compressed packet and feedback transmissions. Here, the indicator function $\phi(i)$ represents the decompression success of the i^{th} packet as

$$\phi(i) = \begin{cases} -1, & \text{decompression of the } i^{th} \text{ packet failed} \\ 1, & \text{otherwise.} \end{cases} \tag{5}$$

For our measurements we shall assume cost symmetry for the sending on the upstream channel of the decompressor entity and receiving on the downstream channel. This assumption also presumes that the loss of individual packets does not affect the overall utility and these cases are handled by the application layer. This consideration is fair, since we only focus on RTP compression for our model and these streams almost exclusively rely on UDP as their transport layer protocol. In practice, application examples can be found in the domains of real–time audio–video transmission, like VoIP calls and video–surveillance. Consequently, we set a

and b in Equation 4 to 1.0. This simplifies the determination of the utility function to

$$U_i' = \sum_{i=0}^{N} (\phi(i) \cdot SH_i - FB_i). \tag{6}$$

We should add that this assumption is scenario–specific and represents the best–case transmission conditions, which might not be attainable in every setup. The round–trip time in cellular networks could be significantly longer and most feedback would arrive much later than in the optimal case, which might result in consecutive decompression failures (due to decompressor desyncronisation).

3 Description of the Procedure

In order to create the input needed for the compression, we have systematically produced a number of compressed streams for our evaluation setup. Let ΔID denote the IPv4 Identification Number difference between consecutive packets, ΔTS denote the RTP Timestamp difference, with other stream characteristics denoted in a similar fashion. We will determine stream types to be used for training different header field dynamics features in our models, as described in Table 1. Note that these streams represent various field changes individually and do not necessarily model real–life audio streams. We will refer to these as training streams, which tell us how the compressor implementation reacts to changes in the input.

Table 1: Systematically generated RTP streams employed for the training of the regression models.

Ideal	Optimal input stream with ΔID = 1, ΔSN = 1, ΔTS = 1 for both IPv4 and IPv6 versions.
Marker–bit	RTP Marker–bit bursts (at the start of talkspurts), as in the ideal scenario, but the Marker–bit is set (unset) every 10 packets for both the IPv4 and IPv6 versions.
Payload type	Represents changes in the RTP Payload type after every 10 packets for both the IPv4 and IPv6 versions.
SSRC	RTP SSRC switches between two specific values after every 10 packets for both the IPv4 and IPv6 versions.
ΔID	IPv4 ID field fluctuations.
ΔTS	RTP Timestamp field fluctuations.
ΔSN	RTP Sequence Number field fluctuations.
ΔM	RTP Marker–bit field fluctuations.
ΔPT	RTP Payload Type field fluctuations.

For the streams that teach header field fluctuations, we created separate versions with 5 different fluctuation probabilities (where $P_f := 0.0, 0.15, 0.25, 0.35$ and 0.5). We also examined different discrete configurations for the building of the models, namely the optimistic mode repetition count in the interval of $[0; 5]$. Each generated stream observes the logical requirements of RTP/UDP/IP traffic, meaning that, for instance, the IP ID or the RTP Sequence Number always increases. The fluctuation probability values are based on an uncorrelated model and combined with the channel loss simulation, along with the above mentioned compressor configurations, provide the complete set of parameters needed for the generation of the training data and the configuration of the measurement runs.

In addition to the above streams, real VoIP streams were captured as well, which we used for the validation of the generated models. For this purpose, we utilised the Asterisk VoIP server[2], which connected a fixed desktop client and an Android smartphone, both using the ZoIPer VoIP client software[3]. This configuration employed the GSM 06.10 codec, which is the full–rate audio codec version that results in 33 bytes of payload. The RTP Timestamp and Sequence numbers have a constant delta of 160 and 1, respectively. The RTP Marker–bit is always set to 0, except for the first two packets. We also employed the Ekiga open source softphone software[4] for further verification as well.

During the assessment, we measured the compression output for each of the above streams under simulated loss rates $P_l \in [0.0; 0.5)$, based on correlated loss probabilities. The classical Gilbert–Elliot model was employed to simulate these correlated losses under dynamically changing channel conditions. Losses are presumed to cover both missing and corrupted packets (we will assume that the receiving device can detect bit–errors and will discard such packets). The Gilbert–Elliot model, depicted in Figure 1, is a two–state Markov–chain, where we simplify the model as follows: (i) no errors occur in the good state and (ii) no packet is conveyed successfully in the bad state.

In our measurement setup we streamline this in such a way that only two parameters are used, i.e., $p_{g,g} := 1.0 - p_{g,b}$ and $p_{b;b} := 1.0 - p_{b,g}$. We initially assume an error–free channel with $p_{g,b} := 0.0$, which we continuously increase with a delta of 0.01. The transition from the bad state to the good state mirrors this configuration, which ensures that the loss pattern will generally contain equally long bursts of packet losses and error–free transmissions. This is sufficient for our measurements, as we are only interested in ascertaining how the compression reacts to continuously repeating losses (bursts) on the channel. Therefore, an exact statistical modelling of a WLAN setup, say, like the ones described in [8] or [9], is outside of the scope of this study.

Together with 10 correlated loss sequences generated for each loss probability instance, resulting in 500 experiments for each configuration/stream combinations, a final data set of 6500 measurement runs with a total of $6,500,000$ packets was created which we preprocessed for further analysis (e.g., calculating the mean sav-

[2]See https://www.asterisk.org for details
[3]See http://www.zoiper.com for details.
[4]See http://www.ekiga.org for details.

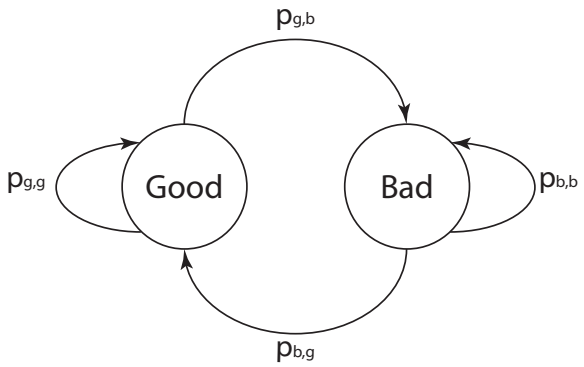


Figure 1: Gilbert–Elliot channel model with two different loss probability states g(ood) and b(ad).

ings and confidence intervals for each measurement). Here, the 6500 runs cover all the relevant *feature–configuration–loss rate* combinations. Though each run contains only 1000 packets, the model would not benefit from a longer stream length as all of the runs simulate just one type of field behaviour. This is sufficient, since if multiple behaviours were to be present in a stream, the feature with the lowest modelled utility would dominate in the output model.

As the input of the machine learning algorithms, we isolated the following independent features:

- *Measurement characteristics*: loss rate
- *Configurations*: optimistic approach repeat count
- *Stream characteristics*: $P_f(\Delta ID)$, $P_f(\Delta SN)$, $P_f(\Delta TS)$, $P_f(\Delta M)$, $P_f(\Delta PT)$, $P_f(\Delta SSRC)$ and IP version

The fluctuation probabilities (P_f) indicate the likelihood, for a given field, that its difference between two consecutive packets will diverge from the established delta value. The expected delta value is well known for certain fields (e.g., IP ID, RTP SN), but for other fields it is possible that a change occurs during the lifetime of a given transmission (e.g., RTP Timestamps). For such cases, we will assume that a constant fluctuation probability that is greater than 0.75 over ten consecutive packets indicates a change in the delta. (We should add that, in general, a fluctuation probability which is greater than 0.5 indicates a change in this delta. However, we decided to employ a greater margin of error to make the transition to a new delta less sensitive). Also, the modelling of other field fluctuations is not required, as they would be classified as *static*, *know* or *inferred*, therefore they either do not change during the lifetime of the given transmission or would belong to separate logical streams.

Overall, our modelling approach attempts to determine the utility function calculated based on Equation 6, which in turn was derived from the measured header

savings (Equation 1) as the dependent feature. To achieve this goal, we chose three different regression methods for the creation of our models and compared them with respect to their attained accuracy. Specifically, we employed Ridge regression, Bayesian linear regression, and Ridge regression with sample weights. In each case the complexity or shrinkage parameter α was set to 0.5, as this turned out to be a good choice regarding accuracy.

We decided to use Ridge regression because not all of the observed features are necessarily independent in a statistical sense or add a significant contribution to the predictor function, as seen in [29]. Ridge regression also uses the L_2 norm to avoid overfitting. We should mention, that based on our observations, a smaller regularisation term like L_1 (LASSO) or $l = 0.5$ – which is used to achieve sparsity in the independent variables – will not result in a significantly better accuracy for any of the models. Also, Bayesian regression was chosen because it avoids overfitting and it is capable of estimating the complexity parameter.

In order to achieve non–linearity in the training input, we used transformations which created polynomial combinations of the input features to the second and third order (i.e., basis functions, as in [1]). We will also provide results with Support Vector Regression (SVR) using RBF kernels as a baseline comparison. The modelling of non–linear dependencies using Neural Networks is outside of the scope of this study.

The machine learning and related techniques were implemented using the *scikit-learn*[5] data mining and data analysis library based on NumPy, SciPy and matplotlib for Python. We also refer to [1] for further details on these methods.

Since the resulting models are represented by a linear function, predictions with the final model are not computationally demanding and can readily be implemented in compressors even on resource–constrained systems.

For the weighted Ridge regression model, we employed weights as shown in Figure 2. A weight set to 100 was assigned to the ideal input stream.

This approach ensures that the model is less biased towards streams with fluctuations. We additionally assigned weights based on the observed loss rate, which was multiplied by the repeat count configuration value of the given measurement. The reason for the latter is that, based on our observations, the predictions become less accurate as the repeat count increases. By assigning higher weights this way, our model will be more reliable in general. The weight function, not taking into account stream types and configurations, is

$$w'(l) = 0.5 \cdot (l + 0.25)^3, \tag{7}$$

where $w'(l)$ denotes the weight value for a given loss rate l in $[0.0, 1.0]$. As the input weight value for the regression, we normalised w' for the $[0.0, 0.5]$ interval like so:

$$w(l) = 1.0 - \frac{w'(l)}{w'(0.5)}. \tag{8}$$

[5]See http://scikit-learn.org.

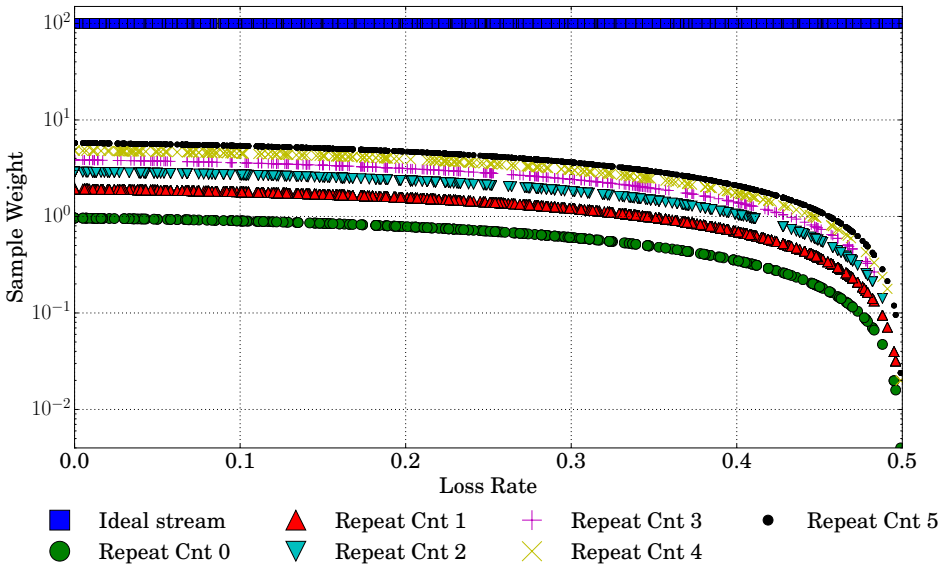


Figure 2: For the weighted Ridge–approach, sample weights are assigned to the observations based on the type of stream being used, the repeat count configuration of the compressor and the (simulated) loss rate found during the measurement stages.

This approach ensures that the model converges less rapidly towards the higher loss rates and will be more accurate for lower ones. Note that weights are only employed during the training of the models and are not part of the input of the verification dataset.

4 The Evaluation of the Prediction Quality

Next, we will make use of the coefficient of determination, denoted by r^2, for the evaluation of the accuracy scores of the models, which tells us how well the data matches a statistical model. An r^2 value of 1.0 indicates that the model perfectly fits the data, while an r^2 of 0.0 indicates that it does not fit the data at all.

We divided our observed measurement datasets into two, depending on whether the streams were the synthetic ones used during the training of the model or those generated by VoIP transmissions (denoted as *training*, *Asterisk* and *Ekiga* in our tables, respectively). The latter ones were used for independent validation. We should mention that we did not use cross–validation techniques on our input dataset (i.e., training), as we only used it to "profile" the compressor implementation. Hence the training accuracy scores serve only as a baseline comparison. Generally speaking, one is only interested in the modelling of the captured real–life streams

(i.e., the Asterisk and Ekiga streams in this study) based on the previously modelled compressor. Since both of these streams are independent of the training dataset (which was created by a packet generator tool), they can capture the accuracy of the model quite well.

4.1 Accuracy

We will first look at the overall r^2 and mean square error results got for the models trained on up to 15 % packet loss rates for both RoHCv1 and RoHCv2, as show in Table 2. We observe that for the training datasets (based on the systematically generated traffic), the Bayesian regression is the most accurate and surpasses the other models by about 10 % at most closely followed by the non–sample–weighted Ridge regression. Also, the weighted Ridge model is on average 15 % worse than the others. This is expected, since the regression takes into account some of the streams with higher precedence, thereby it will not fit as well streams that have lower sample–weights assigned. Consequently, the overall r^2 scores are lower for the weighted Ridge models applied on the training dataset. We also see similar results for the mean square error values in Table 2b.

However, for the independent VoIP streams for version 2 of RoHC the weighted Ridge approach provides the best accuracy which may be about 0.2 better in some cases. Yet, for version 1, the predictions have a very low accuracy in general and relatively high errors, which indicates that RoHCv1's utility function cannot be accurately derived in the $[0\%; 15\%)$ loss–rate range (at least for the implementation in question).

In Table 3, we provide the predictor scores for the models trained on up to 50 % packet losses. We once again observe, that for the training dataset, the Bayesian regression is the most accurate, while for the independent VoIP streams it is the weighted Ridge approach followed by the Bayesian approach. Nevertheless, we observe that all of the models lose at least $\frac{1}{3}$ of their accuracy compared to the 15 % models. However, the results for an SVR model are also provided here as a baseline comparison. In almost every case the various regression approaches outperform the SVR in accuracy by up to 40 %.

Table 3b also shows the calculated mean square error values, which measure the difference between the observed value and the estimated value. We immediately notice that Bayesian regression gives the smallest determined error values for the training input (0.022 for v1 and 0.015 for v2), while the general and the weighted Ridge regression approaches give the smallest errors for the real–world VoIP streams (0.006 for v1 and 0.009 for v2).

We attribute this finding to the construction of our measurement scenarios. Since the weighted Ridge regression is biased towards normal behaviour, it is less likely to capture the erratic fluctuations of the streams used for the training of the regression models (i.e., the unweighted Ridge and Bayesian regressions evaluated here). This makes the utility prediction of the weighted model for the VoIP model significantly more accurate.

Table 2: Utility model accuracy up to 15 % loss–rates.

Stream	*Order*	*Ridge*		*Bayesian*		*Weighted*		*SVR*	
RoHC Version		v1	v2	v1	v2	v1	v2	v1	v2
Training	First	*0.61*	*0.69*	**0.61**	**0.7**	0.46	0.63	0.69	0.72
	Second	*0.8*	*0.8*	**0.86**	**0.87**	0.72	0.72		
	Third	*0.88*	*0.87*	**0.93**	**0.94**	0.8	0.8		
Asterisk	First	*0.4*	*0.68*	**0.41**	0.66	0.39	**0.7**	0.4	0.29
	Second	**0.43**	0.57	*0.43*	*0.59*	0.4	**0.73**		
	Third	*0.35*	0.52	0.28	*0.69*	**0.4**	**0.73**		
Ekiga	First	*0.23*	*0.84*	**0.23**	0.81	0.2	**0.93**	0.35	0.29
	Second	**0.29**	*0.64*	*0.23*	0.66	0.2	**0.94**		
	Third	**0.29**	0.56	0.08	*0.86*	*0.18*	**0.94**		

(a) r^2 values

Stream	*Order*	*Ridge*		*Bayesian*		*Weighted*		*SVR*	
RoHC Version		v1	v2	v1	v2	v1	v2	v1	v2
Training	First	*.062*	*.027*	**.061**	**.027**	.1	.036	.05	.025
	Second	*.032*	*.019*	**.023**	**.012**	.046	.026		
	Third	*.02*	*.011*	**.011**	**.006**	.032	0.018		
Asterisk	First	.023	.016	*.022*	*.014*	**.01**	**.006**	.009	.019
	Second	*.01*	.012	**.006**	*.007*	.01	**.005**		
	Third	**.007**	.009	*.008*	*.006*	.01	**.005**		
Ekiga	First	**.05**	**.008**	*.055*	*.01*	.125	.037	.049	.023
	Second	**.059**	**.017**	*.095*	.037	.125	*.035*		
	Third	**.082**	**.027**	*.124*	.038	.126	*.036*		

(b) Mean Square Error values

4.2 Complexity

We now turn to the question of the overall complexity required for solving the predictor function of the models based on the number of features and summations in the resulting polynomial equations, as suggested in Table 4.

We initially observe that as the order of the solution increases, the coefficient count of our model exponentially increases as well. However, if only coefficients which are larger than 0.01 are taken into account, the feature count may be significantly reduced by about 50 %.

Figure 3 shows the relationship between the order of the polynomial basis functions and the r^2 values relative to the measured loss rates for the unweighed Ridge regression, for both RoHCv1 and RoHCv2.

Table 3: Utility model accuracy up to 50 % loss–rates.

Stream	*Order*	*Ridge*		*Bayesian*		*Weighted*		*SVR*	
RoHC Version		v1	v2	v1	v2	v1	v2	v1	v2
Training	First	*0.66*	*0.61*	**0.66**	**0.61**	0.6	0.55	0.75	0.72
	Second	*0.85*	*0.8*	**0.87**	**0.84**	0.82	0.75		
	Third	*0.9*	*0.87*	**0.92**	**0.9**	0.87	0.84		
Asterisk	First	**0.66**	**0.24**	*0.66*	*0.24*	0.59	0.20	0.63	0.35
	Second	0.58	0.42	*0.59*	*0.43*	**0.62**	**0.45**		
	Third	0.63	0.39	*0.65*	*0.4*	**0.65**	**0.45**		
Ekiga	First	**0.49**	*0.29*	*0.49*	0.29	0.49	**0.51**	0.22	0.36
	Second	*0.48*	0.53	0.48	*0.57*	**0.59**	**0.69**		
	Third	0.55	0.5	*0.57*	*0.6*	**0.64**	**0.71**		

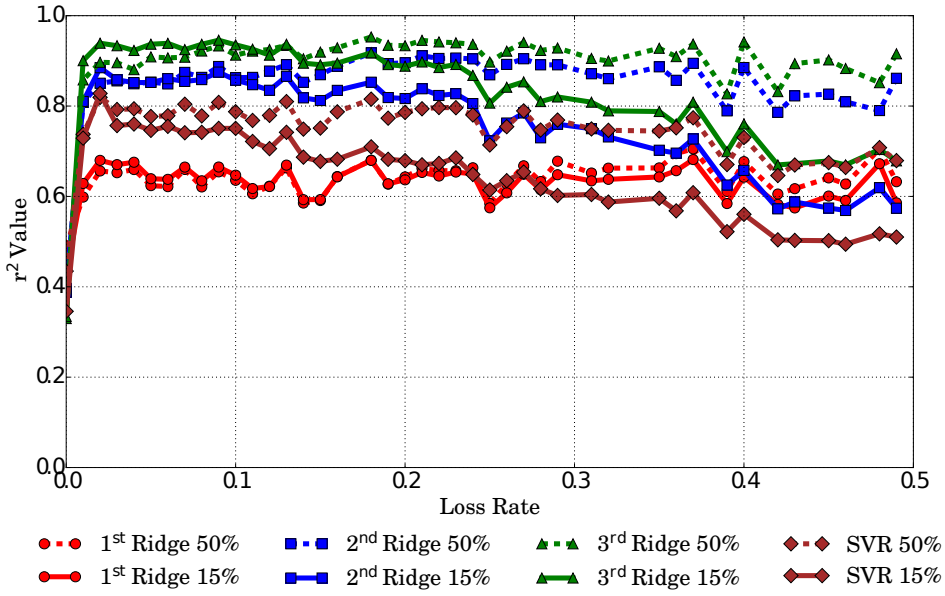
(a) r^2 values

Stream	*Order*	*Ridge*		*Bayesian*		*Weighted*		*SVR*	
RoHC Version		v1	v2	v1	v2	v1	v2	v1	v2
Training	First	*.06*	*.037*	**.06**	**.037**	.08	.046	.044	.027
	Second	*.027*	*.019*	**.023**	**.015**	.032	.024		
	Third	*.019*	.012	**.014**	**.009**	.023	.015		
Asterisk	First	*.028*	.03	**.027**	*.029*	.035	**.019**	.026	.025
	Second	**.027**	.018	*.027*	*.014*	.033	**.013**		
	Third	**.025**	*.015*	*.029*	.015	.031	**.013**		
Ekiga	First	**.039**	**.01**	*.041*	*.012*	.108	.032	.048	.015
	Second	**.057**	**.017**	*.082*	*.031*	.111	.034		
	Third	**.078**	**.029**	*.104*	.039	.111	*.035*		

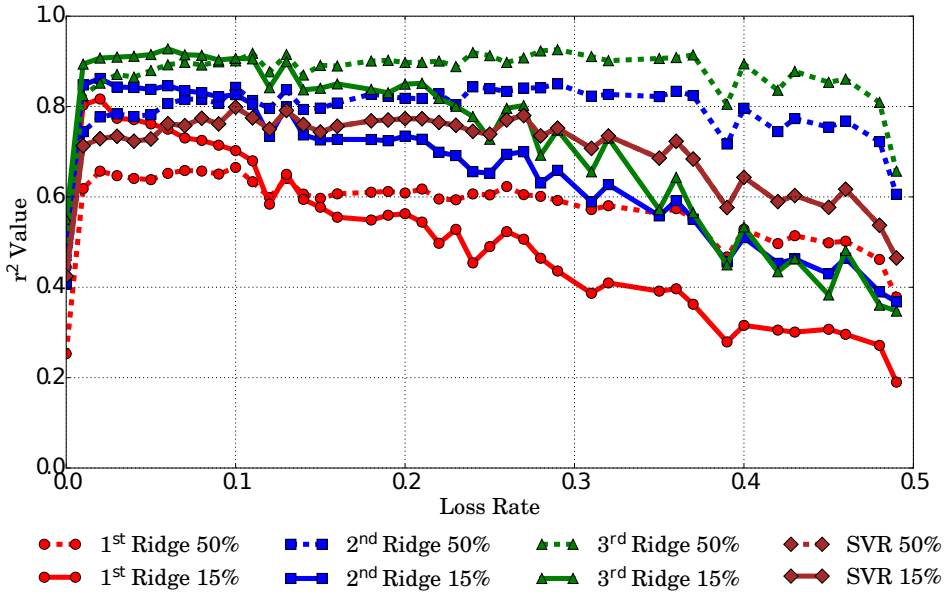
(b) Mean Square Error values

Table 4: The complexity of the models based on the independent feature counts used with the total number of summations (in parentheses) in the predictor equation and the number of features (summations) that have coefficients greater than 0.01.

Order	*All coefficients*	*Coeff's > 0.01*
First	9 (9)	9 (9)
Second	90 (99)	57 (66)
Third	495 (594)	192 (249)



(a) RoHCv1



(b) RoHCv2

Figure 3: 1^{st}, 2^{nd} and 3^{rd} order Ridge regression and SVR accuracy scores got for the r^2 values relative to the loss–rate.

We observe that while the third order solution gives the best results, the accuracy of the second order one is also quite close to that of the third order one. In general, the second order solution results in an increase of r^2 by about 0.1–0.2 compared to the first order solution, whereas the third order r^2 is about 5 % better than the second order solution in some cases. A fourth order solution, however, does not bring about any improvement in the accuracy of the model.

4.3 Accuracy and Losses

We shall now evaluate the model accuracies for the prediction of the real–life VoIP calls for different loss rates. Figure 4a illustrates the relation between the r^2 value and the loss rates of the Asterisk VoIP downstream for RoHCv2.

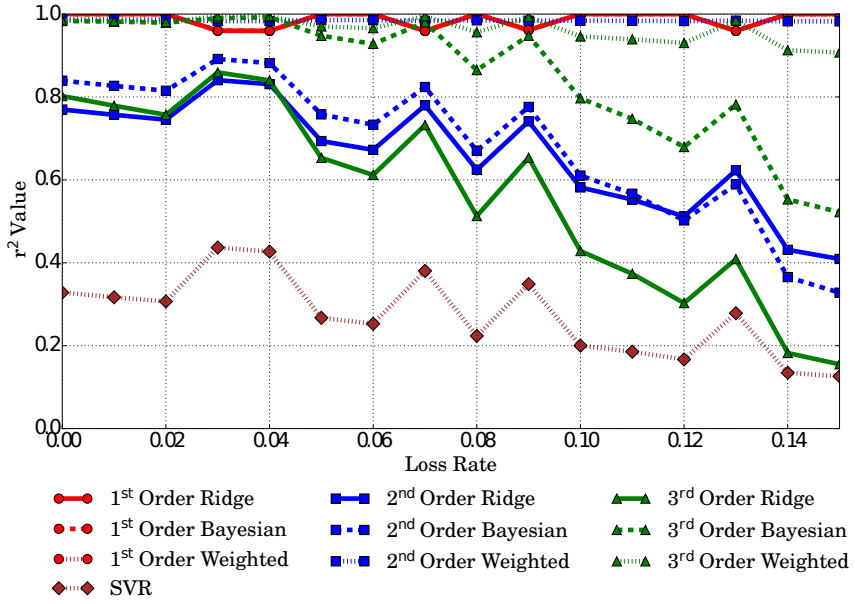
At first glance we notice that all of the first order predictors have a very high accuracy throughout. This is most likely due to this particular stream being well behaved, as in the original stream no significant fluctuations are present. For the downstream Asterisk call (not shown in the figure), the utility function starts to fluctuate quite rapidly from 8 % losses. We also see a similar trend for the Ekiga stream compression (Figure 4b).

For most of the other streams with an increase in the loss rate, the prediction accuracy falls quite rapidly, but in each case the weighted approach proves to be more stable and the accuracy score is about 90 % on average.

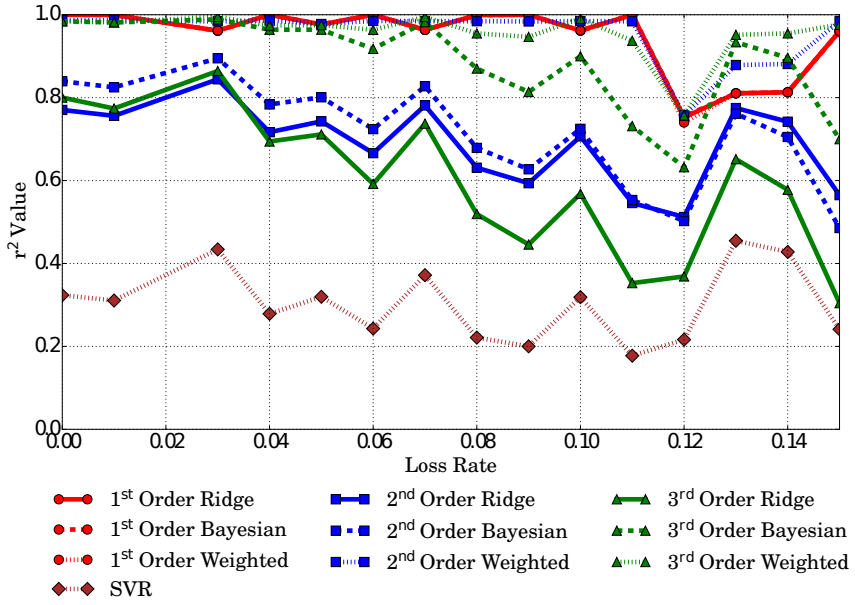
As the order of models increases, the prediction accuracy increases as well. In general, the higher order models provide a more stable prediction and the different model types remain close to each other with an increase in the order. A counter–intuitive result here is that the first order models perform better for higher loss probabilities than the more complex solutions (except for the weighted Ridge approach). For version 1 of RoHC, predicting the utility function is less reliable as the accuracy varies between 0.2 and 0.9 without any recognisable pattern. However, the r^2 scores do become unstable for losses 12 % and higher with both the Asterisk and the Ekiga streams.

5 Conclusions

Here we present a procedure that can be used for profiling Robust Header Compression (version 1 and version 2) compressor implementations. We compared Ridge and Bayesian regressions, Support Vector Regression, as well as Ridge regression with sample weights to construct a predictor in order to derive the compression utility values for RoHCv1 and RoHCv2. This will allow compressors to dynamically adapt to varying channel conditions in the future.



(a) Asterisk



(b) Ekiga

Figure 4: VoIP downstream prediction accuracy with up to 15 % losses for RoHCv2.

Based on the performance evaluation of the applied prediction, we find that the Bayesian approach predicts both the training dataset and the independent VoIP scenarios quite well and it has an r^2 value of 0.66–0.93 and 0.61–0.94 for RoHCv1 and RoHCv2 training streams, respectively. We also reviewed prediction accuracies for real–life VoIP transmissions, which yielded r^2 values of 0.5 on average for both versions.

However, in order to make the predictor less sensitive to a subset of the training streams and also more accurate for the lower loss rates, we devised a weighting strategy for Ridge regression, which is more precise for less error prone channels. This approach achieves a prediction accuracy of at least 0.5 on either of the VoIP transmissions and it may be as good as 0.94 in some cases.

We also find that the second order polynomial base functions provide a good compromise between accuracy and complexity. A second order predictor would require 99 summations, while a third order predictor needs 594. The significantly reduced number of prediction variables and the linear regression make this a suitable approach for low computational resource environments with desired high throughput. When encountering significant channel impairments or larger numbers of streams, however, the reduced prediction accuracy will only allow for a general approximation of the utility level. This issue could be mitigated with a larger and more diverse set of profiles as well as more intricate and computationally demanding approaches to the prediction, which is part of our ongoing work.

Acknowledgment

The authors thank acticom GmbH, especially Gerrit Schulte, for the support and software reference implementations of RoHC 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] Bishop, Christopher M. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[2] C. Bormann, et. al. RObust Header Compression: ROHC: Framework and four profiles: RTP, UDP, ESP, and uncompressed. *Request for Comments 3095*, 2001.

[3] Calveras, A., Arnau, M., and Paradells, J. An Improvement of TCP/IP Header Compression Algorithm for Wireless Links. *Third World Multiconference on Systemics, Cybernetics and Informatics (SCI99) and the Fifth International Conference on Information Systems Analysis and Synthesis (ISAS99), IEEE, Orlando, USA*, vol. 4:pp. 39–46, July/August 1999.

[4] Calveras, A. and Paradells, J. TCP/IP Over Wireless Links: Performance Evaluation. *48th Annual Vehicular Technology Conference VTC 98 IEEE, Ottawa (Ontario), Canada*, vol. 3:pp. 1755–1759, May 1998.

[5] Casner, S. and Jacobson, V. Compressing IP/UDP/RTP Headers for Low-Speed Serial Links. *Request for Comments 2508*, 1999.

[6] Chen, W.-T., Chuang, D.-W., and H.-C.Hsiao. Enhancing CRTP by Retransmission for Wireless Networks. *Proceedings of the Tenth International Conference on Computer Communications and Networks*, pages pp. 426–431, 2001.

[7] Fitzek, F. H., Hendrata, S., Seeling, P., and Reisslein, M. Header Compression Schemes for Wireless Internet Access. *Electrical Engineering & Applied Signal Processing. CRC Press*, page Ch. 10, 2004.

[8] Hartwell, J. A. and Fapojuwo, A. O. Modeling and characterization of frame loss process in IEEE 802.11 wireless local area networks. In *IEEE 60th Vehicular Technology Conference, 2004. VTC2004-Fall. 2004*, volume 6, pages 4481–4485 Vol. 6, Sept 2004.

[9] Hasslinger, G. and Hohlfeld, O. The Gilbert-Elliott Model for Packet Loss in Real Time Services on the Internet. In *14th GI/ITG Conference - Measurement, Modelling and Evalutation of Computer and Communication Systems*, pages 1–15, March 2008.

[10] Hermenier, R. and Kissling, C. Optimization of robust header compression for aeronautical communication. In *Integrated Communications, Navigation and Surveillance Conference (ICNS), 2013*, pages 1–11, April 2013.

[11] Jacobson, V. *Compressing TCP/IP Headers for Low-Speed Serial Links*, February 1990. RFC1144.

[12] Jonsson, L-E. and Pelletier, G. *RObust Header Compression (ROHC): A Compression Profile for IP*, June 2004. RFC3843.

[13] Kim, J., Woo, H., Lee, H., and Lee, M. Dynamic adjustment of optimistic parameter of ROHC for performance improvement. In *2009 International Conference on Information Networking*, pages 1–3, Jan 2009.

[14] Minaburo, A., Nuaymi, L., Singh, Kamal Deep, and Toutain, L. Configuration and analysis of robust header compression in UMTS. In *Personal, Indoor and Mobile Radio Communications, 2003. PIMRC 2003. 14th IEEE Proceedings on*, volume 3, pages 2402–2406 vol.3, Sept 2003.

[15] Minaburo, A. C., Singh, K. D., Toutain, L., and Nuaymi, L. Proposed behavior for robust header compression over a radio link. In *Communications, 2004 IEEE International Conference on*, volume 7, pages 4222–4226 Vol.7, June 2004.

[16] Pelletier, G. *RObust Header Compression (ROHC): Profiles for User Datagram Protocol (UDP) Lite*, April 2005. RFC4019.

[17] Pelletier, G. and Sandlund, K. RObust Header Compression Version 2 (ROHCv2): Profiles for RTP, UDP, IP, ESP and UDP-Lite. *Request for Comments 5225*, 1997.

[18] Pelletier, G., Sandlund, K., Jonsson, L-E., and West, M. *RObust Header Compression (ROHC): Profiles for User Datagram Protocol (UDP) Lite*, July 2007. RFC4996.

[19] Perkins, S. J. and Mutka, M. W. Dependency Removal for Transport Protocol Header Compression over Noisy Channels. In *Proceedings of IEEE International Conference on Communications (ICC)*, pages 1025–1029, Montreal, Canada, 1997.

[20] Ribičre, M. and 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.).

[21] Rossi, M., Giovanardi, A., Zorzi, M., and Mazzini, G. Improved header compression for TCP/IP over wireless links. *Electronics Letters*, vol. 36(no. 23):pp. 1958–1960, November 2000.

[22] Rossi, M., Giovanardi, A., Zorzi, M., and 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.

[23] Seeling, P., Reisslein, M., Fitzek, F.H.P., and 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*, volume 3, pages 1346–1350 vol.3, Dec 2003.

[24] Svanbro, K., Hannu, H., Jonsson, L.-E., and 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.

[25] Tömösközi, M. Robust Header Compression in Wireless Networks. TDK Scientific Students' Association Conference at Budapest University of Technology, 2013. [Online]. Available: http://tdk.bme.hu/VIK/DownloadPaper/Robust-Header-Compression-vezetek-nelkuli (current Aug. 2016.).

[26] Tömösközi, M., Seeling, P., Ekler, P., and Fitzek, F. H. P. Performance Evaluation and Implementation of IP and Robust Header Compression Schemes for TCP and UDP Traffic in the Wireless Context. In *Engineering of Computer Based Systems (ECBS-EERC), 2015 4th Eastern European Regional Conference on the*, pages 45–50, Aug 2015.

[27] Tömösközi, M., Seeling, P., Ekler, P., and Fitzek, F. H. P. Efficiency Gain for RoHC Compressor Implementations with Dynamic Configuration. In *VTC2016-Fall Workshop on Cellular Internet of Things - Emerging Trends and Enabling Technologies*, Sept 2016.

[28] Tömösközi, M., Seeling, P., and Fitzek, F. H. Performance evaluation and comparison of RObust Header Compression (ROHC) ROHCv1 and ROHCv2 for multimedia delivery. In *Workshops Proceedings of the Global Communications Conference, GLOBECOM*, pages 1346–1350, Atlanta, GA, USA, 2013.

[29] Tömösközi, Máté, Seeling, Patrick, Ekler, Péter, and Fitzek, Frank H.P. Performance Prediction of Robust Header Compression version 2 for RTP Audio Streaming Using Linear Regression. In *European Wireless 2016 (EW2016)*, Oulu, Finland, May 2016.

[30] Tömösközi, Máté, Seeling, Patrick, Ekler, Péter, and Fitzek, Frank H.P. Regression Model Building and Efficiency Prediction of RoHCv2 Compressor Implementations for VoIP. In *2016 IEEE Global Communications Conference: Mobile and Wireless Networks (Globecom2016 MWN)*, Washington, USA, December 2016.

[31] Wang, B., Schwefel, H. P., Chua, K. C., Kutka, R., and Schmidt, C. On implementation and improvement of robust header compression in UMTS. In *Personal, Indoor and Mobile Radio Communications, 2002. The 13th IEEE International Symposium on*, volume 3, pages 1151–1155 vol.3, Sept 2002.