Reliable Base Proposal for Header Compression

Máté Tömösközi[1,3], Daniel Lucani[2], Frank H. P. Fitzek[1], Péter Ekler[3]
[1]Communication Networks Group, Technische Universität Dresden, Germany
[2]Department of Engineering, Communications Systems, Aarhus University, Denmark
[3]Department of Automation and Applied Informatics, Budapest University of Technology and Economics, Hungary
Email: mate.tomoskozi@tu-dresden.de, daniel.lucani@eng.au.dk, frank.fitzek@tu-dresden.de, ekler.peter@aut.bme.hu

***Abstract*—The upcoming wireless network generation has put a large emphasis on the fulfilment of high reliability constraints. Nonetheless, the trade-off between these and other network aspects, mainly *delay* and *bandwidth*, is a constant optimisational question and a tough challenge. The various employed protocols add certain encapsulation overheads, which albeit necessary, however could potentially be excessive, such as in the case of various *IoT*, and similar applications with small payloads.**

***Header compression* aims to reduce these headers, but a general problem still plagues the standards since their introduction to loss-prone wireless networks, which is the issue of lost context (re)initialisation packets that can make the compression upstart and the transmission of major changes unreliable, slow and costly. In this paper we propose a solution that circumvents some concerns of traditional header compression context initialisation by the employment of *network coding*, which we call the *reliable base proposal* technique. This provides a finely tunable method for balancing *reliability* and *delay* of decompression with *bandwidth gain*. Our results show that both *compression gain* and *reliability* can be increased over the previous standards.**

Keywords: Robust Header Compression, Compression Context, Network Coding, RoHCv2, Latency, Reliability

I. Introduction

Header compression has been an ever present technology since the early times of wireless networks. Initial concepts, like the one from Van Jacobson [1] have only considered wired networks as their primary target. Later development has expanded the core concepts [2] and added solutions for handling losses during a compressed traffic. *Header compression* approaches reduce the protocol encapsulation between two endpoints of a link, where the reduction in protocol overheads relies on the redundancies found in the different headers of individual packets and between consecutive packets belonging to the same *IP*-flow. However, one of the basic issue for the adoption of *header compression* in the wireless has not been satisfiability resolved as of yet.

A common design theme all over the various compression standards is that the compressor, based on the progression characteristics of the consecutive packet's fields, chooses the smallest packet type from a given compressed packet pool which it deems sufficient for the successful transmission of the header fields. The packets of these pools can be roughly classified into two groups: *non-sequential* and *sequential* packet types. The main difference between the two is that while the former ones transmit data that can be considered persistent in the decompressor context, the latter do not, instead they are derived from the context in various ways.

If we take the compressed packet types of *Robust Header Compression version 2* (*RoHCv2*) from *RFC 5225* as an example, the two corresponding groups are – in *RoHCv2* terminology – the *initialisation and refresh* (*IR*) and the *compressed* header formats. The first group, in this case, only contains the *ir* packet (hence the naming), which is used for the context (re)establishment. Our definition, however, separates two classes, the ones that carry persistent context information without which any consecutive packet would not be accurately decompressible from the ones that do not, i.e., the former being of the *non-sequential* and the latter of the *sequential* ones.[1] Therefore *non-sequential* packets would be the *ir*, *co_repair* and *co_common*, while the sequential ones are *pt_0**, *pt_1**, *pt_2**, etc.

One of the issues that affects header compression algorithms in environments where losses can occur is the reliable establishment of decompressor contexts or bases. Since part of the compression gain is achieved through the storage and maintenance of header fields that are constant or seldom changing (such as *IP destination* and *source address*es), a lack of these values on the receiver side would result in an unrecoverable packet. Moreover, a large part of compression algorithms (e.g., *delta compression*, *LSB compression*, *table-based compression*, etc.) rely on the assumption, that an up-to-date reference value exists on the decompressor side. In case this value is not present or too old, the decompression would either be impossible or could produce an incorrect value.

The issue shows itself during the transmission of *non-sequential* packets on a lossy channel. If a given *non-sequential* packet is lost or damaged, the transmitted persistent data is consequently also lost, which results in the immediate loss of synchronism between the specific compressor and decompressor pair. In order to remedy this, *header compression* solutions have opted to retransmit the bases multiple times and [3] has proposed a method which adapts this to the error rate on the channel. In some cases signalling is used to notify the compressor of an out-of-date context, for example [4] defines a method called *adaptive base proposal* which utilises the feedback on a bi-directional *TCP* connection.

These and similar methods have since become standard solutions in the *header compression* literature and the various standards. There is, however, a common drawback, as they potentially reduce the achievable compression gain via the repetition of these – generally quite large – packet types.

We instead utilise *Random Linear Network Coding* (*RLNC*, or simply *network coding*) to isolate and transmit each of the persistent bases or *non-sequential* packets. *RLNC* [5] is a coding technique that has been used in many applications including distributed storage, bandwidth optimisation, encryp-

[1]Not to be confused with the terminology in *RFC 5225*, Section 6.8.2.1. The two definitions share similarities, but the RFC only considers *compressed header format*s.

tion, etc. It is a unique rateless code which provides recoding, making it applicable for delay sensitive multi-hop networks, such as the mesh. It also supports *online* and *systematic* codes which avoid buffering and unnecessary coding, both of them being critical for the fulfilment of latency requirements [6]. Moreover, the authors devised a *seed-based network coding* method in [7] that minimises the coding overhead to 1 byte only, which is a key requirement for the employment of *network coding* together with header compression.

Consequently, every *non-sequential* packet is partitioned into a number of *coded symbols*, each of which are prepended onto the following packets. In case there are losses on the channel, additional redundant symbols are generated for some of the latter messages. Once the decompressor receives enough of these coded symbols, it can decode them and initialise its context. After that, each of the previously received *sequential* packets are fed into the decompressor and are decompressed as normal. This in turn reduces the cost of context (re)establishment and gives a greater flexibility for balancing *bandwidth* and *delay*. In case the *packet loss probability* of the given channel can be derived accurately, one can completely omit any decompressor feedback as well, which enables the compression to function in a completely *unidirectional* way.

The remainder of this paper is structured as follows. In the next section we introduce our evaluation setup. In Section III we define the employed metrics and in Section IV the achievable *savings* are compared with the already established methods. Finally we look at some of the additional features of *reliable base proposal* before we conclude in Section V.

II. Evaluation Environment

In order to evaluate the proposed method, throughout this contribution we look at four distinct compression strategies, which we define as follows:

- *Uncompressed*: We assume an arbitrary *RTP/UDP/IPv4* stream with a total header length of 40 bytes.
- *Headerless*: This represents the theoretical maximum gain, as no protocol encapsulation is used, resulting in headers of 0 lengths, which is quite rare in most cases.
- *Optimal*: This configuration favours the maximisation of the gain over robustness. The compression does not consider any losses on the channel and keeping compressor-decompressor synchronisation is of no concern.
- *Robust*: This mode always sets the repetition count to a value that guaranties a successful context establishment and corresponds to the *optimistic approach* of *RoHCv2*.
- *Reliable*: Our proposed method, which utilises *network coding* for the establishment of new decompressor contexts and the transmission of *non-sequential* packets.

Figure 1 shows the above four scenarios for 20 consecutive packets which simulate the compression of the *uncompressed* headers. In this case the *optimal* scenario would assume a successful context establishment with the first and tenth packets and would transition to a higher state of compression right after. The *robust* method is, however, more pessimistic and will repeat the context establishment two more times each, which results in more bandwidth usage but a higher tolerance to lost packets. Our *reliable* technique spreads the initial context establishing packet from the *optimal* method over several consecutive packets, which results in similarly high error tolerance as with the *robust* approach, but significantly less transmitted bytes (120 bytes vs. 77 bytes, assuming 1 byte of *network coding* overhead).

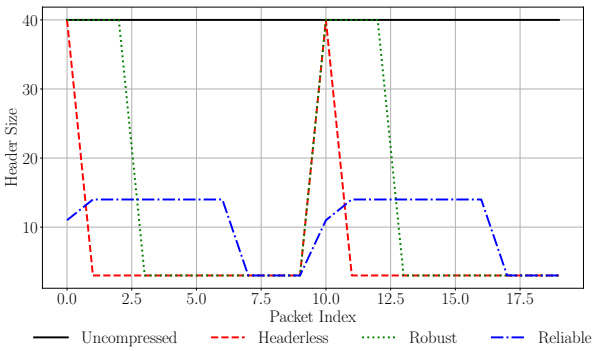

Fig. 1. *IPv4* and compressed header sizes for the *optimal*, *robust* and *reliable* scenarios based on *RoHCv2*.

Note, that in this paper we only assume a *RoHCv2*-like compression in order to evaluate our concept's feasibility for integration with actual compression implementations.

In order to ascertain the reliability of the compression we simulate losses on the channel using correlated loss sequences based on the well known *Gilbert-Elliot model* [8]. This model – even if not resembling any specific wireless system – is well suited for the evaluation of the compression efficiency under a lossy erasure channel as it can produce finely tunable bursty loss sequences. Bursts of packet losses are exactly the condition which could make the compressor lose synchronism with the decompressor, thereby rendering the decompression of any consecutive packet impossible until a complete or partial context refresh arrives.

During the assessment, we focused on simulated loss rates $P_l \in [0.0; 1.0)$, which is derived from the *Gilbert-Elliot* model's overall mean loss probability. These losses are presumed to cover both missing and corrupted packets. The model in question is illustrated in Figure 2 and is a two state Markov-chain, where we simplify the model in such a way that (i) no errors occur in the good state and (ii) no packet is conveyed successfully in the bad state.

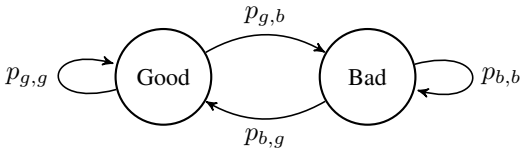

Fig. 2. Gilbert-Elliot channel model with two states g(ood) and b(ad).

In our measurement setup we streamlined even more, so 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. To achieve statistical confidence we evaluate using 10000 distinct loss sequences with 10000 individual packets for each discussed setup.

III. Metrics

In this section we introduce our metrics used for ascertaining the differences between the various scenarios. First, we define *context availability* C similarly to the percentage based ($[0.0, 1.0)$) metric of *service* (or *system*) *availability*. In the original sense, a very close value to 1 would translate to a very low downtime for the given service. For example, a 99.9 % or 0.999 availability refers to about 9 hours of worst case downtime for a system on a yearly basis. In our case of *context availability* a 0.999 value means 1 failed *context establishment* per 1 million packets where *non-sequential* packets occur at an average rate of once in 100 packets. A common way to refer to the order of magnitude is to employ the total number of *nines* in the percentage. In the previous case, 0.999 would translate to 3 n or 3 *nines*. We use this notation interchangeably throughout the presentation of our contribution.

Given a specific *availability* score, determining the minimum number of transmissions to achieve this for the *robust* method (t_{robust}), one needs to solve the following equation:

$$1 - C = \mathbf{p}(\epsilon)^{t_{robust}}, \tag{1}$$

where C is the desired *context availability* and $\mathbf{p}(\epsilon)$ is the *packet loss probability* of the channel. From this the t_{robust} is obtained fairly straightforward as:

$$t_{robust} = \frac{ln(1-C)}{ln(\mathbf{p}(\epsilon))}. \tag{2}$$

To determine the same for the *reliable* method ($t_{reliable}$), we construct a separate *Markov-chain* based on the underlying *Gilbert-Elliot model*. This will consist of $G \cdot 2$ *transient* and one *absorbing* states, where G is the *generation size* of the *network coding* instance. Figure 3 illustrates a chain for $G = 4$.

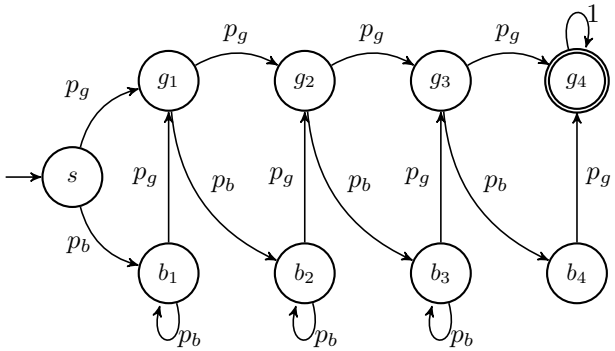

Fig. 3. A sample *Markov-chain* generated in order to determine the *context availability* of the *reliable* method with $G = 4$.

Given an initial state S, the model can transit with appropriate probabilities ($p_g := p_{g,g}$ and $p_b := p_{b,b}$) to a good or bad state ($g_i, b_i : i \in \{1, 2, 3, 4\}$). Once at least G number of "good" transitions with p_g probability have been achieved the model ends up in the *absorbing* state, which is, in this case, g_4. This corresponds to a successful *context establishment* or the successful update of the context with a *non-sequential* packet.

In turn, in order to determine $t_{reliable}$, we calculate the *absorption probability* and solve the following system:

$$\begin{aligned} x &= \mathbf{P}^{t_{reliable}} \\ x &\leq 1 - C, \end{aligned} \tag{3}$$

where $\mathbf{P}$ is the transition matrix of the model. The *absorption probability* after i iterations is obtained as $(\mathbf{P}^i)_{G*2+1,G*2+1}$.

For the evaluation of the compression *savings* or *gain* one can simply measure the ratio of the compressed packet size to the original uncompressed packet. This can be expressed as:

$$S = 1.0 - \frac{\|T_{co}\|}{\|T_{uc}\|}, \tag{4}$$

where $\|T_{co}\|$ and $\|T_{uc}\|$ are the total transmitted bytes with compression and without compression.

In case retransmissions occur more often, we prefer to use the *payload delivery efficiency* E metric in order to ascertain the effort it takes to deliver a given payload successfully to the recipient under losses as:

$$E = 1.0 - \frac{\|L\| + \|F\|}{\|T\|}, \tag{5}$$

which is the ratio between the total received bytes at the sink node that can be delivered to the application layer without errors and the total transmitted bytes including any redundant transmissions. Specifically, $\|L\|$ is the amount of dropped packets in bytes (excluding redundant packets), $\|F\|$ is the amount of packets which failed decompression in bytes and $\|T\|$ the total transmitted bytes of the given scenario.

Moreover, we employ two auxiliary metrics for ascertaining the ratio of *discarded* ($\|F\|$) and *delayed message*s at the receiver to the total number of transmitted ones. The latter ratio refers to packets that cannot be processed at the exact time of reception but were successfully decompressed later on.

IV. Results

On Figure 4 we show the achievable *compression gain*s. As expected, the *context availability* is inversely proportional to the *header savings*, as either the context establishment is repeated more times or more redundant symbols are sent. However, in all scenarios the *reliable* approach is better or as good as the *robust* mode. Moreover, increasing the *availability* is more costlier for the *robust* approach than for the *reliable*, especially for high *loss rate*s.

Note that the *gain* for the *reliable* method flattens out after about 80 % losses. This is due to the rate at which *non-sequential* packets are sent (in this case after every $IR = 100$ packets), which acts as the high bound of the given scenario, as we do not consider consecutive base establishments. Meaning, that in case there are not enough *sequential* packets to successfully transmit at least *G* of them with a specific *C availability*, that context is considered not initialised and all information is discarded. This can be, however, remedied with relative ease.

The trade-off for the increased *compression gain* is the added *delay* until the decompession context is initialised and is

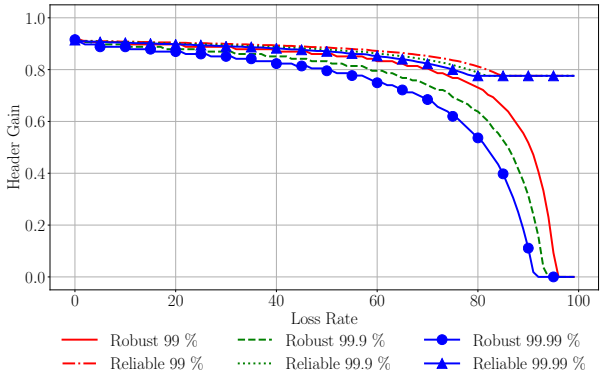

Fig. 4. *Compression gain* of the *robust* and *reliable* methods for *context availability* rates of 2, 3 and 4 nines against the correlated *loss probabilities*.

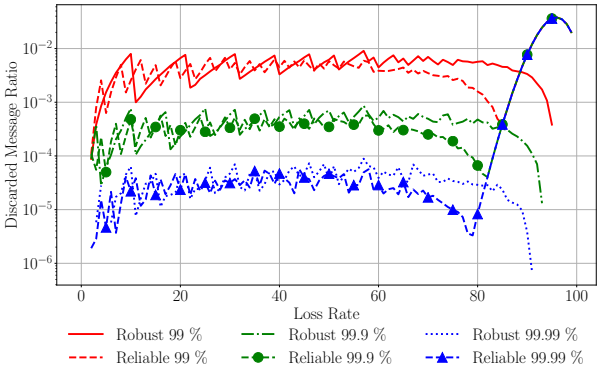

Fig. 6. Ratio of successfully received but discarded compressed packets of the *robust* and *reliable* methods for *context availability* rates of 2, 3 and 4 nines against the correlated *loss probabilities*.

ready to decompress *sequential* packets. As seen in Figure 5, an increase of 1 n generally adds 1.5 times more delayed packets and is proportional to the observed loss rate increasing linearly from about 0.07 to 0.15 for 99 % reliability and more than 0.22 for 99.99 %. At around 80 % losses the delayed message ratio falls rapidly toward 0. This occurs at the same position and for the same reason as in Figure 4, since failed context initialisations result in discarded packets instead of delayed ones.

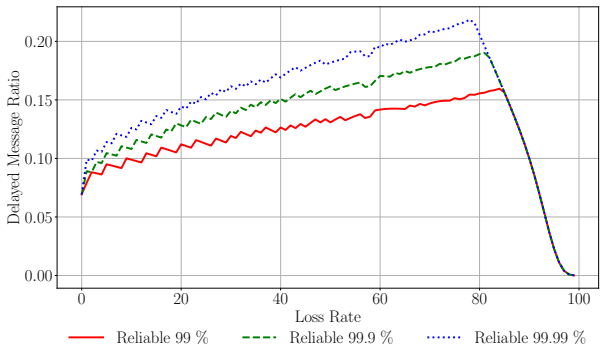

Fig. 5. Ratio of delayed messages to the total transmitted ones for the *robust* and *reliable* methods with *context availability* rates of 2, 3 and 4 nines against the correlated *loss probabilities*.

Figure 6 illustrates the number of packets discarded by the decompressor due to contexts being out of date. The plot indicates that the *reliable* method successfully recovers packets that would be otherwise discarded by the *robust* technique. With the increase in availability one improves the *robust* method drastically, but considering the cost for high loss rates, one can see the benefit of employing the *reliable* method.

The number of network coded symbols is limited by the *context refresh rate* (denoted by *IR*, an $IR = 5$ means that every fifth packet is enforced to be a *non-sequential* or in *RoHCv2* terms, an *ir* packet) and the number of redundant packets the compressor has to send in order to overcome the losses. Of course, more frequent refreshes mean that one receives more benefit from the *reliable* context establishment – which is precisely the opposite of what one observes when utilising the *robust* technique – but this is limited by the number of packets sent between refreshes, as before. Given a *generation size* of 8 symbols, a *context availability* of 2 nines and a refresh interval of $IR = 50$ packets, it is not hard to see that, at precisely 71 % loss rate, the compressor would need to send 51 symbols, which is one more than the rate of refreshes (see Figure 7). This means, that at 71 % losses and higher, one cannot guarantee the two nines of *availability* with this configuration. Consequently, finding the right balance between *availability*, *generation size* and *context refreshes* is important. Note, that since in this scenario there are no major retransmissions, a small increase of *efficiency* can be considered significant in contrast to some of the authors' other works, such as [7].

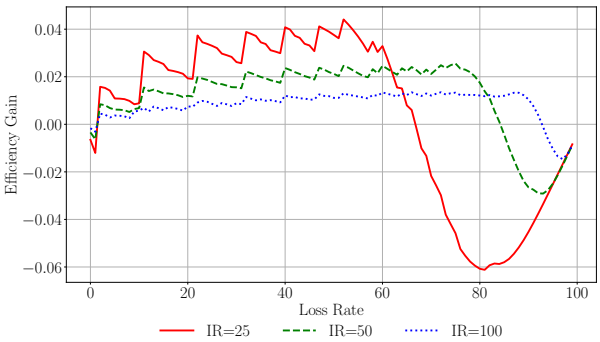

Fig. 7. *Payload efficiency* gain of the *robust* and *reliable* methods for *context availability* rates 2 nines against the correlated *loss probabilities*, 40 bytes of payload and varying repetition intervals with *generation size* 8.

Based on our observations so far, we endeavour the construction of an arbitrary scenario where we balance the costs and gains of the various configurations in order to find the most efficient one. To achieve this we define a *penalty function* – where negative values are considered rewards – for each of the following aspects:

- *Reliability penalty*: Exponentially decreases by $p_r = (1 - C)^2 \cdot 2 - 1$, meaning that reliability values close to 1 get a bonus, while values under ∼0.7 get an increasing penalty.
- *Compression gain penalty*: Exponentially decreases similarly to the above with a higher rate by $p_g = 1 - S^3$,

but without bonuses ($p_g \in [1, 0]$), which increase quite rapidly as the gain diminishes. The motivation behind this is that one should always strive to achieve the highest *gain* possible, which is usually around 0.9.

- *Delay penalty*: Exponentially increases by $p_d = D^4 \cdot 2 - 1$, where the shortest delays yield a bonus and ones longer than 80 packets get penalties. Since we assume that $IR = 100$, the delays which get close to 100 have to be avoided, otherwise failed context initialisations become common.

Based on the penalties we execute a *brute force* lookup on a pre-generated table, which finds an optimally balanced solution from two directions given a specific *loss rate* that achieves the highest *availability*: $min(D)$ gets the shortest *delay*, while $max(R)$ results in the highest *availability*.

Figure 8 presents the best settings chosen by the algorithm with the above defined penalty functions. We see that minimising for $max(R)$ results in a generally small *generation size*, which is a bit worse than the corresponding *robust* solution by about 25 packets. However, the *reliable* method increases both the *gain* and the *reliability* by a few points. When we limit $min(D)$ to at least 2 nines ($R \leq 0.99$) we achieve results that are on average 1 nine better.

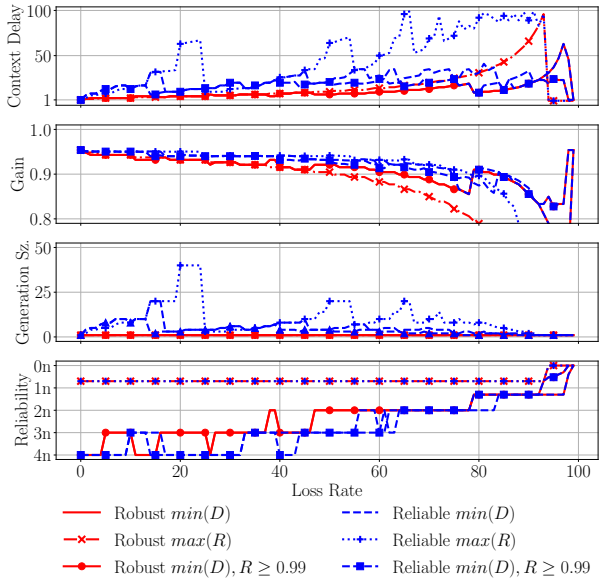

Fig. 8. Optimal settings for the *reliable base proposal* and the *robust* methods employing the *penalty functions*.

In turn, finding the maximum *availability* while keeping the shortest delay results in a gradually increasing *delay* and higher *compression gain*s for the *reliable* technique, while the *availability* is locked in at about 0.7 for both (this is where the penalty transitions from positive to negative values).

V. Conclusions

Fifth generation applications, amongst other things, will rely on the fulfilment of high reliability constraints. This, however, sometimes happens at the expense of latency and bandwidth usage. *Header compression* has been used extensively and successfully for the reduction of the packetisation overhead commonly observed in *IP* networks. Nonetheless, this is sometimes achieved by rendering the fulfilment of *delay* and *reliability* needs as secondary concerns. Consequently, when the right balance between these three aspects has been found, one can utilise *header compression* at its utmost benefit.

A serious problem that affects header compression techniques is the issue of lost context (re)initialisation packets which can make the compression unreliable. In this paper we propose a solution that circumvents the problems of traditional header compression context initialisation by the employment of *random linear network coding*, which we call the *reliable base proposal* technique. This provides a finely tunable method for balancing reliability of decompression and bandwidth gain during compression. We show that both *compression gain* and *context availability* can be increased over the previously establish standard approaches, while keeping the *delay* costs acceptable and minimal. It also enables reliable compression on a unidirectional channel at a never before seen efficiency.

Currently we are working on a more integrated solution with actual compression implementations and real-life streams to thoroughly show the benefits of our new concept and with more intricate integration of *network coding*.

Acknowledgments

The authors would like to thank Maroua Taghouti of the Technische Universität Dresden for her support in some of the theoretical aspects of this paper. This work was supported by the BME-Artificial Intelligence FIKP grant of EMMI (BME FIKP-MI/SC), the János Bolyai Research Fellowship of the Hungarian Academy of Sciences and the SCALE-IoT Project (Grant No. 7026-00042B) granted by the Danish Council for Independent Research, the Aarhus Universitets Forskningsfond Starting Grant Project AUFF-2017-FLS-7-1, and Aarhus University's DIGIT Centre.

References

[1] V. Jacobson, "Compressing TCP/IP headers for low-speed serial links," *Request for Comments 1144*, 1990.

[2] M. Degermark, M. Engan, B. Nordgren, and S. Pink, "Lowloss tcp/ip header compression for wireless networks," *Wireless Networks*, vol. 3, pp. 375–387, Oct 1997.

[3] A. Calveras, M. Arnau, and J. Paradells, "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. vol. 4, pp. pp. 39–46, July/August 1999.

[4] A. Giovanardi, G. Mazzini, M. Rossi, and M. Zorzi, "Improved header compression for tcp/ip over wireless links," *Electronics Letters*, vol. 36, pp. 1958–1960, Nov 2000.

[5] T. Ho, M. Medard, R. Koetter, D. R. Karger, M. Effros, J. Shi, and B. Leong, "A random linear network coding approach to multicast," *IEEE Transactions on Information Theory*, vol. 52, pp. 4413–4430, Oct 2006.

[6] M. Tömösközi, F. H. P. Fitzek, D. E. Lucani, M. V. Pedersen, and P. Seeling, "On the delay characteristics for point-to-point links using random linear network coding with on-the-fly coding capabilities," in *European Wireless 2014; 20th European Wireless Conference*, pp. 1–6, May 2014.

[7] M. Tömösközi, D. E. Lucani, F. H. Fitzek, and P. Ekler, "Unidirectional robust header compression for reliable low latency mesh networks," in *2019 IEEE International Conference on Communications (ICC): Mobile and Wireless Networks Symposium (IEEE ICC'19 - MWN Symposium)*, (Shanghai, P.R. China), May 2019.

[8] P. Sadeghi, R. A. Kennedy, P. B. Rapajic, and R. Shams, "Finite-state markov modeling of fading channels - a survey of principles and applications," *IEEE Signal Processing Magazine*, vol. 25, pp. 57–80, Sep. 2008.