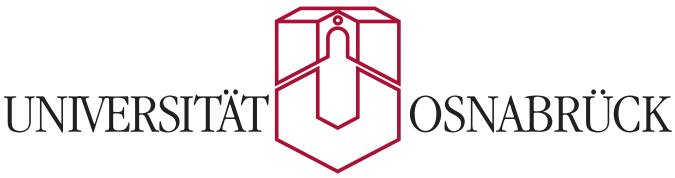


Bridging the Gap: Integration, Evaluation and Optimization of Network Coding-based Forward Error Correction

Dissertation
(kumulativ)

zur Erlangung des Doktorgrades (Dr. rer. nat.)
des Fachbereichs Mathematik/Informatik
der Universität Osnabrück

Autor:
Bertram Schütz

Osnabrück, 03.06.2021

1. Gutachter:	Prof. Dr. rer. nat. Nils Aschenbruck Universität Osnabrück
2. Gutachter:	Prof. Dr. rer. nat. Frank Kargl Universität Ulm
Vorgelegt am:	03.06.2021
Verteidigt am:	27.07.2021

"It doesn't matter how beautiful your theory is [...], if it doesn't agree with experiment, it's wrong. In that simple statement is the key to science."

Richard Feynman (Erice, 1964)

Abstract

The formal definition of network coding by Ahlswede et al. in 2000 has led to several breakthroughs in information theory, for example solving the bottleneck problem in butterfly networks and breaking the min-cut max-flow theorem for multicast communication. Especially promising is the usage of network coding as a packet-level Forward Error Correction (FEC) scheme to increase the robustness of a data stream against packet loss, also known as intra-session coding. Yet, despite these benefits, network coding-based FEC is still rarely deployed in real-world networks. To bridge this gap between information theory and real-world usage, this cumulative thesis will present our contributions to the integration, evaluation, and optimization of network coding-based FEC.

The first set of contributions introduces and evaluates efficient ways to integrate coding into UDP-based IoT protocols to speed up bulk data transfers in lossy scenarios. This includes a packet-level FEC extension for the Constrained Application Protocol (CoAP) [P1] and one for MQTT for Sensor Networks (MQTT-SN), which levels the underlying publish-subscribe architecture [P2]. The second set of contributions addresses the development of novel evaluation tools and methods to better quantify possible coding gains. This includes *link 'em*, our award-winning link emulation bridge for reproducible networking research [P3], and also *SPQER*, a word recognition-based metric to evaluate the impact of packet loss on the Quality of Experience of Voice over IP applications [P5]. Finally, we highlight the impact of padding overhead for applications with heterogeneous packet lengths [P6] and introduce a novel packet-preserving coding scheme to significantly reduce this problem [P4]. Because many of the shown contributions can be applied to other areas of network coding research as well, this thesis does not only make meaningful contributions to specific network coding challenges, but also paves the way for future work to further close the gap between information theory and real-world usage.

Zusammenfassung (German Abstract)

Obwohl die formale Definition von Network Coding durch Ahlswede et al. im Jahr 2000 zu entscheidenden Durchbrüchen in der Informationstheorie geführt hat, wird Network Coding bisher in der Praxis selten eingesetzt. Beispielsweise löst Network Coding zwar die Bottleneck-Problematik in Schmetterlings-Netzwerken und bricht das Min-Cut Max-Flow Theorem für Multicast, dennoch hat sich die Technik noch nicht in realen Netzwerken etabliert. Dabei bietet besonders der Einsatz als Fehlerkorrekturverfahren auf Paketebene zur Steigerung der Robustheit gegenüber Paketverlusten vielversprechendes Potential in drahtlosen Szenarien. Um diese theoretischen Vorteile wirklich nutzbar zu machen, schlägt die vorliegenden kumulative Dissertation deshalb die notwendige Brücke zwischen Informationstheorie und Praxis. Dazu werden die geleisteten wissenschaftlichen Beiträge und Publikationen in den Bereichen Integration, Evaluation und Optimierung von Network Coding-basierter Fehlerkorrektur zusammengefasst, diskutiert und in einen gemeinsamen Kontext gestellt.

Der erste Bereich dieser Beiträge befasst sich mit der Integration von Network Coding in UDP-basierte IoT Protokolle, um die Dauer von Over-the-Air-Updates in verlustbehafteten Szenarien zu reduzieren. Speziell werden dazu zwei Fehlerkorrektur-Erweiterungen vorgestellt und evaluiert, eine für das Constrained Application Protocol (CoAP) [P1] und eine für MQTT for Sensor Networks (MQTT-SN) [P2]. Die zweite Säule dieser Arbeit beinhaltet die Entwicklung neuer Methoden zur Evaluation der potentiellen Network Coding Vorteile. Dies umfasst *link 'em* [P3], eine Layer-2 Link Emulation Bridge für reproduzierbare Netzwerkforschung, und *SPQER* [P5], eine neue Metrik zur Bewertung des Einflusses von Paketverlust auf die Quality of Experience (QoE) von Voice-over-IP (VoIP) Anwendungen auf Basis automatischer Spracherkennung. Die dritte und finale Beitragsreihe befasst sich mit dem Thema Padding Overhead für Anwendungen mit heterogenen Paketlängen. Dazu analysieren wir das Auftreten dieses Overheads für mehrere echte Anwendungen [P6] und stellen ein neues Coding Schema vor, welches den Overhead massiv reduziert, ohne die grundsätzlichen Paketstrukturen zu verändern [P4]. Durch die Summe an wissenschaftlichen Beiträgen in den drei Kategorien löst die vorliegende Dissertation nicht nur spezifische Probleme im Bereich Network Coding, sondern legt zusätzlich den Grundstein für kommende Arbeiten um die Lücke zwischen Informationstheorie und Praxis weiter zu schließen.

Contents

List of Publications

[P1] Bertram Schütz, Nils Aschenbruck
"Adding a Network Coding Extension to CoAP for Large Resource Transfer", 41st IEEE Conference on Local Computer Networks (LCN), Dubai, UAE, November 2016.
DOI: 10.1109/LCN.2016.122

[P2] Bertram Schütz, Jan Bauer, Nils Aschenbruck
"Improving Energy Efficiency of MQTT-SN in Lossy Environments Using Seed-Based Network Coding", 42nd IEEE Conference on Local Computer Networks (LCN), Singapore, October 2017.
DOI: 10.1109/LCN.2017.87

[P3] Bertram Schütz, Stefanie Thieme, Nils Aschenbruck,
Leonhard Brüggemann, Alexander Ditt, Dominic Laniewski, Dennis Rieke
"Link 'em: An Open Source Link Emulation Bridge for Reproducible Networking Research", International Conference on Networked Systems (NetSys), Munich, Germany, March 2019.
DOI: 10.1109/NetSys.2019.8854509

[P4] Bertram Schütz, Nils Aschenbruck
"Packet-Preserving Network Coding Schemes for Padding Overhead Reduction", 44th IEEE Conference on Local Computer Networks (LCN), Osnabrück, Germany, October 2019.
DOI: 10.1109/LCN44214.2019.8990879

[P5] Bertram Schütz, Nils Aschenbruck
"SPQER: Speech Quality Evaluation using Word Recognition for VoIP Communication in Lossy and Mobile Networks", IEEE Open Journal of the Computer Society, July 2020.
DOI: 10.1109/OJCS.2020.3011392

[P6] Bertram Schütz, Nils Aschenbruck
"The Impact of Bit Errors on Intra-Session Network Coding with Heterogeneous Packet Lengths", Symposium of the 45th IEEE Conference on Local Computer Networks (LCN), Sydney, Australia, October 2020.
DOI: 10.1109/LCNSymposium50271.2020.9363259

1 Introduction

Despite its theoretical benefits, e.g., breaking the min-cut max-flow theorem for multicast communication [19] or solving the coupon collector problem for content distribution [15], network coding has not yet emerged as a stable technique in real networks. Some open problems still have to be solved to bridge the gap between information theory and real-world usage. The presented thesis contributes to this goal by solving existing challenges and progressing the overall state-of-the-art of network coding-based Forward Error Correction (FEC).

In general, network coding can be distinguished into two different categories, inter-session and intra-session network coding. While both approaches use the same underlying Galois Field theory to encode packets in form of linear combinations, they aim for different benefits. The traditional inter-session coding approach combines packets across multiple data streams to solve routing problems and is typically used to increase the overall throughput in multicast or broadcast communication. A prominent example for inter-session coding is Ahlswedes butterfly network [1], where an intermediate node does not simply forward the received packets but sends linear combinations instead to efficiently utilize the capacity of a bottleneck link. Intra-session coding on the other hand is used to increase to robustness of a data stream between two nodes against packet loss. This is done by injecting additional encoded redundancy packets to enable packet-level Forward Error Correction (FEC). Thus, intra-session coding can be interpreted as solving a system of independent linear equations. If a sufficient number of packets are received, the whole system can be solved and all original packets can be restored. Thereby, it does not matter exactly which packet was received or lost. The so increased robustness is especially helpful in mobile and lossy scenarios, where packet loss occurs frequently due to interferences or signal degradation.

It has to be noted that the term intra-session network coding sometimes leads to confusion. While both types of network coding use similar Galois Field operations, the original idea of network coding was not to inject redundancy for robustness, but combine packets to overcome information flow limitations. Due to these roots, some of the initially developed formal fundamentals and proofs are only valid for inter-session coding, e.g., one core assumption of the work by Ahlswede et al. [1] is that all transmissions happen instantly. Thus,

for clarity, we use the term network coding-based FEC here instead of intra-session network coding. A more detailed explanation of both network coding types, their distinctions, and relation to other FEC coding approaches, will be presented in Section 2.1.

1.1 Overall Goal & Main Contributions

The overall goal of this thesis is to improve the state-of-the-art of network coding-based FEC to bring the theoretical benefits to practice. Thus, the guiding motto of this work is to bridge the gap between information theory and real-world usage, which is visualized in Figure 1.

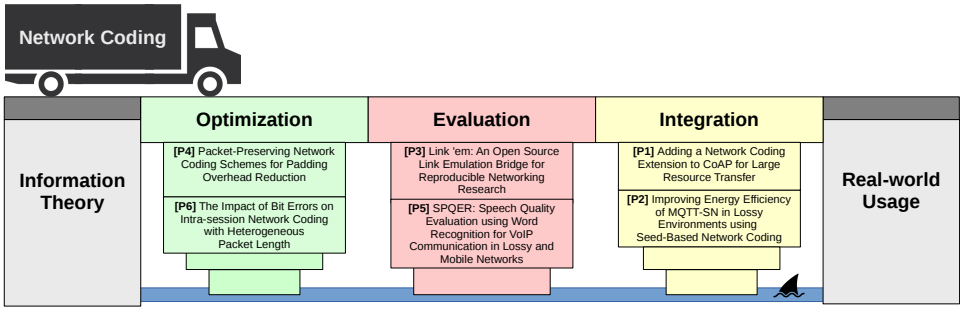


Figure 1: Thesis motto: "Bridging the gap" between information theory and real-world usage for network coding-based FEC.

In order to reach this goal, we made meaningful contributions to three different research areas: the integration, evaluation, and optimization of network coding-based FEC. This cumulative thesis summarizes and discusses our contributions, which are essentially the following:

- **Integration** of network coding-based FEC for reliable file transfer within UDP-based Internet of Things (IoT) protocols, namely a coding extension for the Constrained Applications Protocol (CoAP) [P1] and one for MQTT for Sensor Networks (MQTT-SN) [P2].
- Development of novel **evaluation** tools and methods to better quantify possible coding gains. These are *link'em* [P3], a raspberry Pi-based layer-2 link emulation bridge for reproducible networking research, and *SPQER* [P5], a machine learning-based metric for VoIP Quality of Experience evaluation in lossy networks.

- Coding overhead analysis for real-world applicationss and codec **optimization** for padding overhead reduction of data streams with heterogeneous packet lengths. This includes a novel packet-preserving coding scheme [P4] and also an evaluation of the impact of bit errors on potential coding gains [P6].

The connections between these contributions are visualized in Figure 2. We will further dissected this figure in section 3, when linking our publications and presenting the context of each one to highlight its place within this thesis.

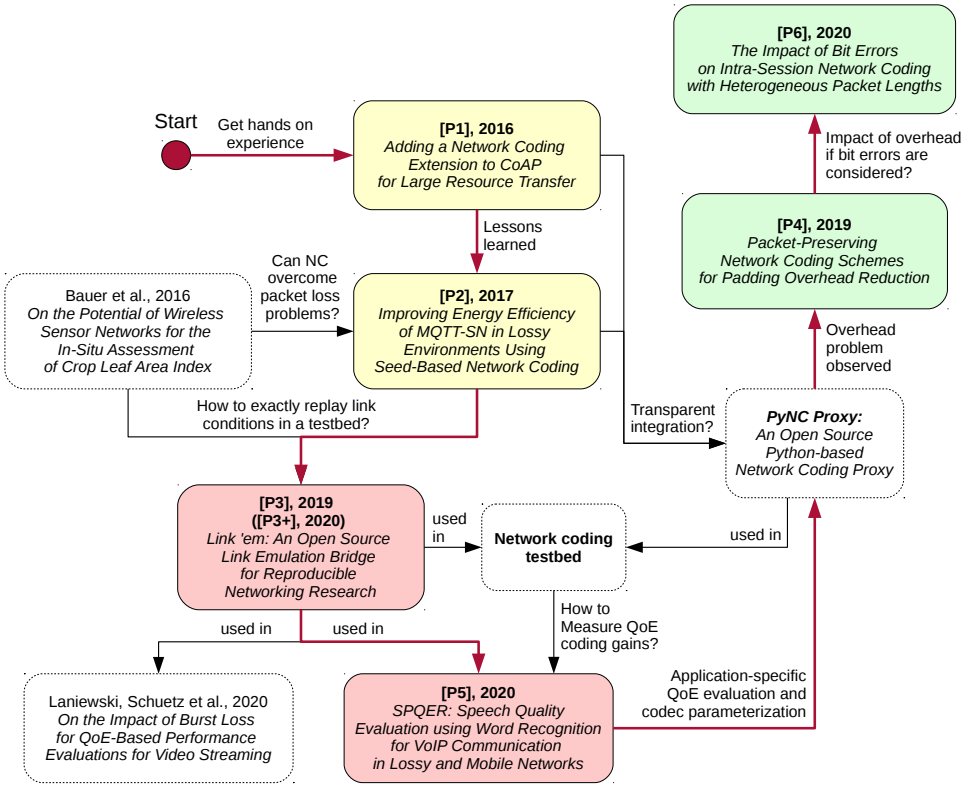


Figure 2: Road map linking our contributions and publications. This thesis' structure is highlighted by the red path.

1.2 Thesis Structure

The structure of this thesis does not directly follow the chronological order of our publications but is laid out in a more content-oriented form to better show the connections between them. As highlight by the red path in Figure 2, we first present our publications towards the integration of network coding-based FEC, then show to the development of new evaluation methods, before finally covering the topic of codec optimization. Sticking to our *bridging the gap* analogy (c.f. Figure 1), we start on the practical side of network coding and move from integration to evaluation, to finally arrive on the more formal side of codec optimization.

More precisely, the remainder of this thesis is structured as follows: Following this introduction, Section 2 summarizes the necessary basics and related work. Section 3 then contains the three main subsections of this thesis, which are our contributions towards the *Integration*, *Evaluation*, and *Optimization* of network coding-based FEC. After summarizing and discussing the contributions in each research area, Section 4 concludes the overall achievements of this thesis and presents further challenges for future work. Finally, the full texts of all our publications are attached.

2 State-of-the-art & Related Work

2.1 Network Coding in a Nutshell

The following section gives a short and comprehensive overview about the network coding basics. It is based on our explanation in [45] and follows the notations by Sundararajan et al. in [49]. More in-depth introduction to network coding can be found in [17]. The main idea of network coding is to treat packets not as a sequence of immutable bytes but transform them into algebraic symbols. This allows to add, subtract or multiply packets, and especially to form and solve equations. The concept was firstly introduced by Ahlswede et al. using a simple XOR coding [1], and later extended by Li et al. to Linear Network Coding (LNC)[33]. Finally, Ho et al. showed that the coefficients of these linear combinations can be chosen at random, while still achieving linear independence of the encoded packets with high probability [19], giving Random Linear Network Coding (RLNC) its name.

Algebraic Transformation: In network coding theory a packet is treated as a vector over an extended binary Galois Field $GF(2^q)$. The term Galois field is used in honor of french mathematician Évariste Galois, who has made major contributions to the field of abstract algebra by connecting field theory and group theory. A commented version of Galois' original paper is published in [11]. The smallest viable Galois field, $GF(2^1)$ contains only the elements 0 and 1. Thus, each field element represents one bit. By increasing the field size, larger data types can be represented. In general, an element of $GF(2^q)$ represents q bit of data. For example, an element of $GF(2^2)$ represent 2-bits. In this work, we use a field size of 2^8, which is recommended in [17], because most networking operations work on a byte-level. With this approach, a byte can be treated as a septic polynomial (degree of seven) over the chosen field. For example, the 8-bit sequence 11001010 can be interpreted as an element of $GF(2^8)$, by treating it as the polynomial $1x^7+1x^6+0x^5+0x^4+1x^3+0x^2+1x+0$. Hence, each element of $GF(2^8)$ represents a specific byte of data. Using this transformation from the physical into an algebraic domain, an original (uncoded) packet p of length l is not interpreted as a chain of bytes anymore but as a concatenation of l $GF(2^8)$ elements, which can be written as:

$$p_1 = [p_{11}, p_{12}, \ldots, p_{1l}] \; with \; p_{11}, \ldots, p_{1l} \in GF(2^8) \tag{1}$$

It has to be noted that different fields of the same size can be created, depending on the chosen generator polynomial. For the purpose of network coding it is irrelevant which polynomial was used. It is only important that all nodes use the same field, i.e., a field of the same size with the same generator polynomial. We omit here further details about the creation and operation definition of Galois Fields, but concentrate on the relevant information regarding the usage for network coding. More formal information and explanations can be found in [47]. What makes Galois fields especially interesting for coding theory is their property of being closed against multiplication, addition, subtraction and division. Thus, any operation performed on elements from the field again returns an element from the same field. This property can be visualized using an addition table, as shown in Figure 3 for $GF(2^2)$.

+	0	1	2	3
0	0	1	2	3
1	1	0	3	2
2	2	3	0	1
3	3	2	1	0

Figure 3: Exemplary addition table of $GF(2^2)$.

In Galois field arithmetics, addition is similar to an element-wise XOR operation. This can be verified by looking at the shown addition table of $GF(2^2)$. For example, converting element 3 to binary (11) and adding element 2 (10) results in element 1 (01), similar do applying a bit-wise XOR. Because the field is closed against such an operation, the result is again an element of the field. This property is crucial for coding theory, and network coding in particular. Adding or multiplying two elements results in an element of the same size. Thus, forming a linear combination of two packets results in an encoded packet of the same length.

Encoding & Decoding: Based on this Galois theory, an encoded packet q_i can be created by forming a linear combination of original packets. For example, a coded packet q_i can be created as a linear combination of the original packets p_1, p_2 by using the coding coefficients $\alpha_{1i}, \alpha_{2i} \in GF(2^8)$:

$$q_i = \alpha_{1i} p_1 + \alpha_{2i} p_2 \tag{2}$$

The number of packets, which are combined in the encoding step is called the generation size g. The presented example above uses $g = 2$. Due to the high probability of linear independence when using random coefficients, it is possible to create more than g linear independent packets without a complex code book. The amount of additional packets is called the redundancy rate r. The example below shows the encoding process for traditional full density RLNC with a generation size of $g = 2$ and a redundancy of $r = 0.5$.

$$\begin{aligned} q_1 &= \alpha_{11} p_1 + \alpha_{21} p_2 \\ q_2 &= \alpha_{12} p_1 + \alpha_{22} p_2 \\ q_3 &= \alpha_{13} p_1 + \alpha_{23} p_2 \end{aligned} \tag{3}$$

Upon receiving a new linear independent packet q_i, the receiver adds it to the decoder's storage, forming the decoding matrix D. If the rank $rank(D)$ of this matrix equals g, the receiver can decode the matrix to restore the original packets. Since the coded packets are linear combinations, the decoding process corresponds to solving a system of $rank(D)$ linear equations with g unknowns, which can be solved by using Gaussian elimination. Thus, the decoding condition can be written as: $rank(D) \geq g$.

Because the encoder can produce at least $g + g \cdot r \geq g$ independent encoded packets, RLNC has an implicit packet-level Forward Error Correction (FEC) property. The decoder can restore all original packets, even if some encoded packets get lost, as long as the decoding condition is fulfilled. It is not necessary to send feedback or request the retransmission of a specific packet to achieve reliability. This network coding-based FEC is often referred to as intra-session network coding and differs from the inter-session network coding approach, where packets of different streams and/or senders are combined. While intra-session coding is used to overcome packet loss within one data stream, inter-session coding can be used as an alternative to traditional routing [19], with a prominent example being Ahlswedes butterfly network [1]. This paper focuses on the FEC aspect of intra-session coding, especially on a dedicated coding scheme, called systematic RLNC (sRLNC). The distinct feature of systematic RLNC is the usage of two different encoding phases. The first g packets are send uncoded, which can be interpreted as using an identity matrix to form the first g linear combinations. This so called systematic phase is followed by the redundancy phase, where $g \cdot r$ random linear combinations are created by combining all original packets. An exemplary encoding process for systematic RLNC with $g = 2$ and $r = 0.5$

is shown in the following set of equations:

$$\begin{aligned} q_1 &= 1p_1 + 0p_2 \\ q_2 &= 0p_1 + 1p_2 \\ q_3 &= \alpha_{13}p_1 + \alpha_{23}p_2 \end{aligned} \tag{4}$$

Since the first g uncoded packets are guaranteed to increase the decoders rank upon reception, there is no drawback in using sRLNC as a FEC scheme on the first hop in comparison to traditional RLNC. Upon reception, the decoder can directly make these uncoded packets available to the application layer without the need for other packets to arrive. Thus, invoking less decoding delay than traditional RLNC, where a packet can only be used by an application after the reception of at least $g - 1$ other encoded packets.

The following example demonstrates the usage of sRLNC FEC in a simple end-to-end communication. Assume we have a network consisting of two nodes S and R, which are connected by link L. An application running on S wants to transmit the original packets p_1, p_2, and p_3 to a receiver application on R. Further assume the link is prone to packet loss with packet loss rate PLR_L. To add robustness against this loss, an sRLNC encoder is deployed on S and a corresponding decoder runs on R. This setup is shown in the following figure:

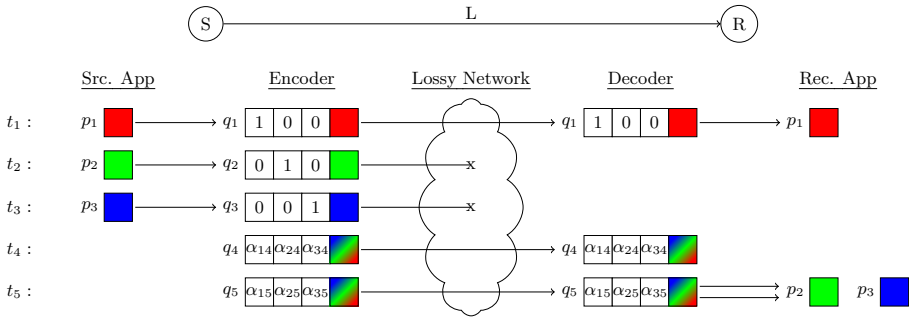


Figure 4: sRLNC FEC example.

Upon receiving q_5, the decoding condition $rank(D) \geq g$ becomes fulfilled. Thus, the decoder can restore p_2 and p_3 and forward them to the receiving application. Despite the losses on the link and without additional feedback or retransmissions, the communication still achieves full reliability. The figure also illustrates the distinct advantage of sRLNC over traditional RLNC.

Because the first three send packets are basically uncoded, the red packet p_1 can directly be forwarded to the receiver without decoding delay.

FEC Capability: In general, the FEC capability of a sRLNC codec depends on the chosen generation size g, redundancy rate r, and the links loss rate PLR_L. When analyzing possible coding gains by calculating the packet loss rate after decoding PER_{DEC}, we must consider if the decoding condition $rank(D) \geq g$ is fulfilled or if this is not the case. In other words, we must distinguish if the subset of received packets contains at least g packets ($rank(D) \geq g$), or if it contains less than g packets ($rank(D) < g$). Starting with the simple case $rank(D) \geq g$, due to linear independence, the decoding matrix is always solvable (or at least with very high probability). Thus, all received subsets, which contain at least g packets result in a PER of zero and must not be considered any further when calculating PER_{DEC}. On the other hand, subsets with $rank(D) < g$ must be analyzed more carefully, depending on their composition. Because an identity matrix is used during the systematic phase of the encoding process, c.f., Figure 4, the first g packets can be treated as being uncoded. The receiver can forward these systematic packets to the application layer without further decoding, even if the decoding condition $rank(D) \geq g$ is not fulfilled. Thus, for subsets with $rank(D) < g$, the number of usable packets is equal to the number of systematic packets, while redundancy packets must be ignored. Formalizing the thoughts above leads to the following equation to calculate the average PER of an sRLNC generation after decoding, denoted as PER_{DEC}:

$$PER_{DEC} = \sum_{k=0}^{g-1} \sum_{i=0}^{g-k-1} \binom{g}{k} \binom{g \cdot r}{i} \left(\frac{g-k}{g}\right) \cdot P(k,i) \tag{5}$$

$$with \quad P(k,i) = (1 - PLR_L)^{k+i} \cdot (PLR_L)^{g \cdot (r+1) - (k+i)} \tag{6}$$

Notice that the factor $P(k,i)$ weights each subset according to its probability of occurrence by calculating the probability to receive exactly k out of g systematic while also receiving i out of $g \cdot r$ redundancy packets. The binomial coefficients additional factor in the amount of all possible combinations with exactly this distribution of packets. As mentioned before, it is not necessary to consider subsets with $rank(D) \geq g$ when calculating PER_{DEC}. Thus, the upper limit of the first sum with index k is set to $g-1$, while $g-k-1$ is used for the second sum, limiting the total cardinality of the analyzed subsets to

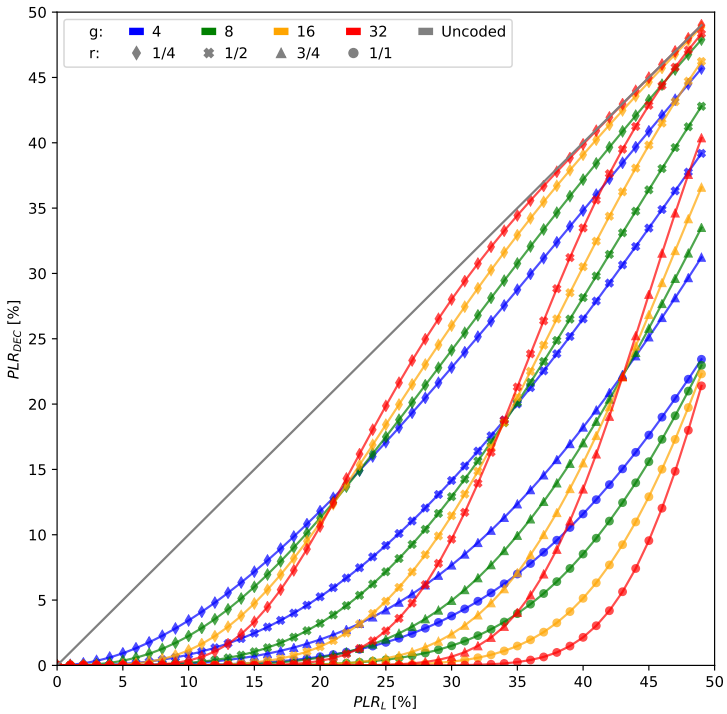


Figure 5: Packet loss rate after decdoing (PER_{DEC}) for sRLNC.

$g-1$. It has to be noted that the shown equation assumes a Bernoulli packet loss distribution, where each packet has a independent loss probability of PLR_L. In reality, packet loss is seldom truly Bernoulli distributed, as we have shown for traces of a WSN deployment [P2]. We will further discuss the occurrence and modeling of packet loss in section 2.3.1. To visualize the the FEC capability of sRLNC, we have implemented equation 7 and run numerical calculations. The result is shown . As shown in in Figure 5, the potential packet loss reduction depends heavily on the combination of the chosen generation size g, redundancy rate r, and link loss rate PLR_L.

To further highlight its FEC benefits, the sRLNC coding gain can be expressed in terms of relative percentage change $\Delta\%PLR_{DEC}$:

$$\Delta\%PLR_{DEC} = \frac{PLR_L - PLR_{DEC}}{PLR_L} \cdot 100 \tag{7}$$

This metric represents the packet loss reduction in percent, if network coding-based FEC is used. For example, applying sRLNC(g=4, r=0.25) to a data

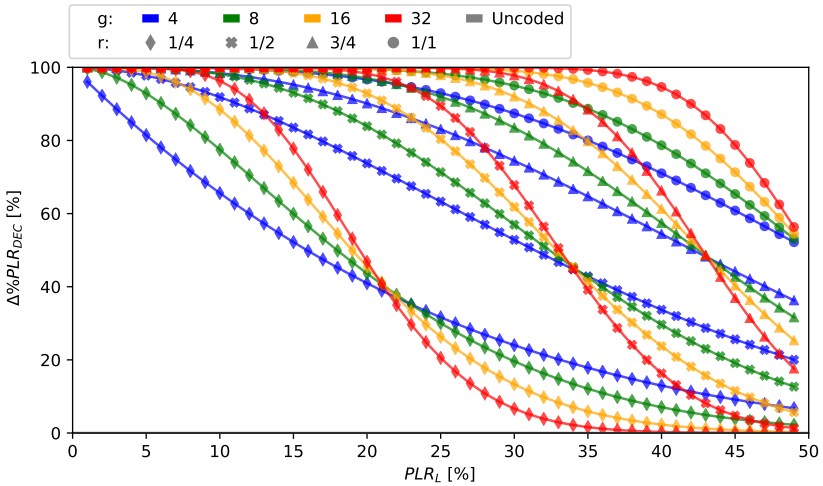


Figure 6: Relative percentage change $\Delta\%PLR_{DEC}$ for sRLNC after decoding.

stream on a link with a native packet loss rate of $PLR_L = 15\%$ results in a packet loss rate after decoding of $PER_{DEC} = 7.18\%$ (c.f., Fig. 5, blue diamonds). This corresponds to a relative PLR reduction of 52.20%. One way to increase the FEC capability is to crank up the redundancy rate. For example, using sRLNC(g=4, r=0.5) reduces the final loss rate to $PER_{DEC} = 2.47\%$, which is a gain of $\Delta\%PLR_{DEC} = 83.52\%$ (c.f., Fig. 5+6, blue crosses). A higher robustness can also be achieved by setting a larger generation size, i.e., sRLNC(g=16, r=0.25), to make the scheme more robust against occasional burst losses. Finally, combining a larger generation size with a higher redundancy yields the best results, i.e., using sRLNC(g=16, r=0.5). Such a parameterization reduces the loss rate to $PER_{DEC} = 0.23\%$, which is a gain of $\Delta\%PLR_{DEC} = 98.48\%$ (c.f., Fig. 5+6, yellow crosses). While this coding gain is significant, it comes at the cost of additional redundancy and a larger decoding delay. Thus, selecting the right parameterization is one important aspect when deploying the codec.

Other FEC codecs: While network coding-based FEC is a rather young research field, the first bit-level FEC code was already introduced by american mathematician Richard Hamming in 1950. His now famous (7,4) Hamming code was initially developed to make the execution of punched paper tape programs robust against bit errors. By adding three parity bits, the code can protect a sequence of four original bits against a single-bit error, thus, the (7,4) notion. In 1960, Irving S. Reed and Gustave Solomon extended this approach to a block-based codec, which enables a more efficient encoding of large data batches. To distinguish network coding-based FEC from these traditional bit-level codes, the following section will first present a short summary of traditional FEC codes, then highlight the differences. In general, error correcting codes are not limited to network communication. Any data transmission, which can be prone to errors and where feedback is not possible or prohibitively expensive, can profit from FEC. One prominent example is the usage of Reed–Solomon codes to make optical disc media, e.g. CDs and DVDs, robust against scratches. For critical, high-value data storage, a similar coding scheme is used for protection against corruption. These Error-correcting code memory (ECC memory) chips are especially relevant for space missions, where bit errors are much more common due to solar flares and can cause significant problems. Over the years, several different coding strategies were developed with special optimizations and goals. For example, low-density parity-check (LDPC) codes [12] are used in the IEEE 802.16 standard, i.e., WiMAX, because they have a very low complexity. Turbo codes [7] on the other hand naturally apply interleaving to increase the robustness against bit error bursts. Further information on the FEC basics and different code types can be found in *"Error Correcting Codes - A Mathematical Introduction"* by D J. Baylis [4]. A more networking-oriented FEC introduction is given in [14].
The main difference of the previously mentioned traditional FEC codes in comparison to network coding-based FEC is the payload level, on which they operate. Hamming codes, Turbo codes, Reed-Salomon codes, LDPC codes, and all related coding strategies work on a bit- oder byte level. Even though interleaving (c.f. [7]) allows to spread the redundancy over a larger sequence of bytes, the encoding itself is still done on a byte-level. Network coding on the other hand directly forms independent linear combinations of packets. Thus, while the other codes can correct bit or byte errors, network coding-based FEC can recover lost packets. This characteristic difference enables several advantages, but also offers new challenges, as will be highlighted in the following chapters.

2.2 Integration

The full potential of network coding can only be put into practice, if the coding scheme is well integrated into the existing network infrastructure. Yet, a sound integration offers unique challenges and is often easier said than done. Thus, this section will present the state-of-the-art of network coding integration and summarize the basic concepts of MQTT-SN and CoAP to give the background for our contributions [P1] and [P2].

2.2.1 Related Integration Approaches

To distinguish our contributions from the related work, the following section highlights the most prominent existing integration approaches, categorized by the network stack layer on which they operate. This is shown in figure 7 using the IETF stack model, with the session, presentation and application layer being merged for clarity. We have further added additional shim layers to categorize cross-layer solutions. Even though the focus of this thesis lies on intra-session coding, notable inter-session coding approaches are also mentioned.

Starting with network coding on the physical layer (PNC), the most prominent and often cited publication is *"Hot topic: Physical-layer network coding"* by Zhang et al. [55]. Instead of mixing bits and bytes, PNC directly superimposes the electromagnetic waves of two signals on one another to form signal-level linear combinations. While PNC is the most drastic way to implement network coding, it also promises very significant gains. For example, in [31], the authors provide an example, which shows a potential throughput gain of 100% for a passive optical network (PON). The approach was refined by Lu et al. in [34] to cope with the asynchrony between transmitted signals. Still, the two main remaining problems of PNC are the complex implementation and the missing compatibility with legacy systems. Mixing two electromagnetic waves and outputting the result requires special hardware and is not possible on regular network cards. Also, seamless interoperability with traditional Ethernet or 802.11 Wi-Fi physical layers is not given. More information on PNC can be found in a survey by Liew et al. [31], which was published in 2013.

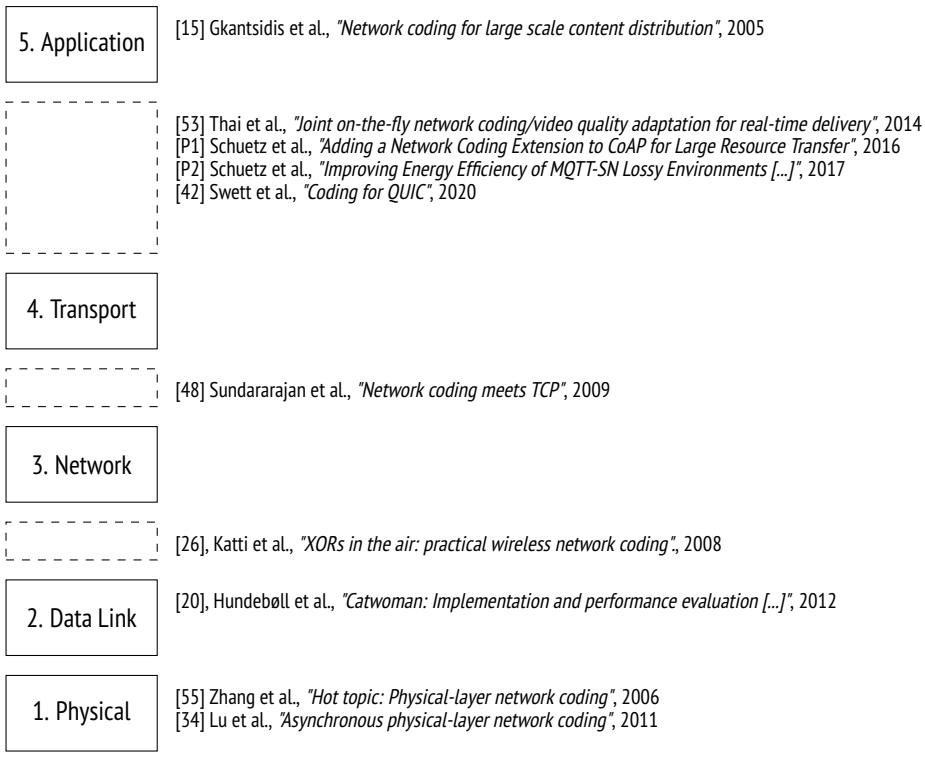


Figure 7: Related integration approaches, categorized by stack layer.

Even before PNC became its own research field, most network coding approaches targeted the data link or network layer. For example, in their groundbreaking work [26], Katti et al. have inserted an XOR inter-session network coding scheme on a shim layer between the network and data link layer, called *COPE*. To send encoded packets downstream, Katti et al. initially wanted to use the broadcast mode of the underlying 802.11 MAC layer. But due to poor reliability and missing back offs, they had to settle for a pseudo-broadcast, which is in reality a modified unicast that can be sniffed by the other nodes. Unfortunately, this approach works only if all devices are set to promiscuous mode, which is not necessarily possible for all consumer-grade network devices. For example, a common smartphone can only overhear other packets if a root kit is installed, e.g., CyanogenMod (discontinued) or its successor LineageOS [1] for android.

[1] https://www.lineageos.org/, accessed 02.08.2020

A similar, yet more practical usable approach, called *CATWOMAN*, was proposed by Hundebøll et al. in 2012 [20]. While the coding itself works similar to *COPE*, *CATWOMAN* operates on top of the already well established mesh routing protocol *B.A.T.M.A.N.*. By utilizing the inherited routing mechanisms, *CATWOMAN* easily detects coding opportunities. The authors claim that this approach achieves a performance gain of up to 60% compared to a pure relaying scheme in a scenario with two bidirectional flows (known as the Alice and Bob scenario). It has to be noted that the implementation uses the advanced version of *B.A.T.M.A.N.*, which switched the routing to layer 2, making *CATWOMAN* essentially a link layer protocol. Yet, the general idea would also work with the old layer 3 version of *B.A.T.M.A.N.*.

The previously mentioned approaches focused on inter-session network coding to solve bottleneck or routing challenges. Ascending the stack naturally leads from routing towards host-to-host communication services. In general, inter-session coding is done low in the stack, while intra-session implementations reside on the upper layers. Thus, network coding-based FEC is typically integrated above the network layer.

Probably the most popular publication of an intra-session coding scheme is Coded TCP (CTCP) by Sundararajan et al. [48]. Prior to the development of CTCP, several studies have shown significant problems of regular TCP when used in lossy networks. Usually, TCP ensures fairness between competing data streams using additive increase/multiplicative decrease (AIMD) congestion-avoidance. This strategy was designed to cope with queue drops in cable-bound networks but not with occasional packet loss due to interference or signal degradation, which can happen in wireless communication. To reduce the amount of packet loss, and, thus, increase the overall throughput, CTCP uses encoded packets in combination with a new TCP acknowledgment strategy. Instead of acknowledging sequence numbers, CTCP checks for degrees of freedom by using the new notion of *seen packets*. This is the main novelty introduced by Sundararajan et al. in [48]. More information and implementation details on CTCP can be found in their follow-up paper *"Network Coding Meets TCP: Theory and Implementation"* [49]. There, it is also mentioned that their initial implementation resides on a shim layer below the transport layer. Thus, we marked it accordingly in Figure 7.

While CTCP has shown great potential, it is essentially a new transport layer protocol and not compatible with other devices in regular 802.11-based networks. This legacy problem is not exclusively new for network coding

protocols, but applies to any new transport layer protocol. Especially the compatibility with middleboxes and firewalls is one crucial limitation. For example, many firewalls include packet filter, which only allow traditional UDP and TCP packets to pass. Other protocols will be dropped, which greatly limits the deployability of a novel transport protocol. This problem was also recognized by Googles development team, when creating the Quick UDP Internet Connections protocol (QUIC) as a faster alternative for web browsing than common TCP over HTTP/2. Already the first paragraph of the official QUIC for Chromium website [2] describes this issue: *"Because TCP is implemented in operating system kernels, and middlebox firmware, making significant changes to TCP is next to impossible. However, since QUIC is built on top of UDP, it suffers from no such limitations.*" So instead of developing a completely new transport protocol, QUIC simply works on top of UDP. A similar approach was implemented for the Stream Control Transmission Protocol (SCTP) by tunneling SCTP over UDP. The corresponding UDP encapsulation was proposed in RFC 6951 [43]. Because such tunneling is necessary and only TCP and UDP are possible candidates, this leads to the question: Which transport protocol is better suited for coding integration, TCP or UDP? In most cases, the answer is simple: UDP. One fundamental principal of TCP is transport layer reliability. Therefore, depending on the TCP flavor, a different type of automatic repeat request (ARQ) retransmission scheme is used. These retransmissions are triggered, before any packet-level FEC capabilities would be applied on the layers above. Thus, TCP is not suited as the underlying transport protocol for network coding-based FEC. UDP on the other hand is often not chosen due to is missing reliability. Yet, for the integration of packet-level FEC, this is not a problem, but an opportunity, and, thus, the main reason why we have selected UDP-based protocols for our coding integration approaches in [P1] and [P2].

A prominent UDP-based FEC empowered protocol is TETRYS, which is a sliding window encoding scheme for real-time critical applications, especially video streaming. The basic idea was published by Thai et al. in [53]. A further definition of TETRYS as an application layer protocol was uploaded in form of an experimental draft [10] of the IETF Coding for efficient NetWork Communications Research Group (NWCRG).

The same working group also currently proposes a draft to add packet-level FEC to QUIC, based on TETRYS' sliding window encoding scheme [42]. QUIC is especially suited for a FEC extension, because is uses UDP as the

[2] https://www.chromium.org/quic, accessed 30.07.2020

underlying transport protocol and also has a partial reliability feature to still guarantee the completeness of a transfer. An early evaluation for different packet loss rates and delays was presented by Michel et al. in [38]. The study showed that the FEC extension can drastically lower the download completion time for short web transfers but can lead to problems for longer downloads or in low-bandwidth configurations.

Finally, there also exist applications with directly integrated coding. The most prominent example, and one of the first commercial applications of network coding, is the peer-to-peer (P2P) content distribution service *Avalanche* [3]. The software was developed by Pablo Rodriguez and Christos Gkantsidis of the Microsoft research team. *Avalanche* works similar to BitTorrent, but instead of just distributing chunks of the original content, the peers store linear combinations of smaller blocks. Without coding, a receiving node needs one copy of each individual chunk to restore the original content. Unfortunately, the probability to receive an independent chunk sinks with each chunk already received. This problem is also known as the coupon collector problem. If network coding is used, each encoded block is linear independent with high probability. Thus, a receiving node just needs a sufficient number of blocks, and not each individual original chunk. This idea is essentially network coding-based FEC on a content level. Gkantsidis et al. presented a performance evaluation in [15], claiming that coding can reduce the file download time by more than 2-3 times. Additionally, the new system is more robustness and better able to handle problematic scenarios where the server and nodes leave the network.

2.2.2 CoAP & MQTT-SN

Due to the aforementioned legacy problems and integration constrains, we focused our integration research on UDP-based protocols. We have chosen to work on coding for IoT devices, because the constrained nodes often have to communicate wireless under harsh conditions with limited power, which makes robustness against packet loss a critical feat. Within the IoT, the two most prominent approaches are the Constrained Applications Protocol (CoAP) and MQTT for Sensor Networks (MQTT-SN). To give the background for our coding extensions, the relevant details of these two protocols will be summarized here.

[3] https://www.microsoft.com/en-us/research/project/avalanche-file-swarming-with-network-coding/, accessed 01.08.2020

CoAP: The Constrained Applications Protocol (CoAP) is featured in our first publication [P1]. The protocol was proposed in RFC 7252 [40] by the IETF Constrained RESTful environments (CoRE) Working Group. It is designed to give constrained devices access to the main features of a representational state transfer (REST) architecture. This includes resource requests via get, put, post and delete, similar to HTTP. There are several design aspects to CoAP, which are helpful to integrate coding. For example, a 2-bit type field T in each message header indicates, if it is confirmable or non-confirmable request. Thus, by setting this field to non-confirmable, it is possible to disable CoAPs ARQ and implement FEC reliability instead, without breaking the standard. Another important design aspect is the extensibility of CoAP messages using option fields. By including an option, the sender can force the receiver to process the containing message in a specific way. CoAP's specification [40] already includes two of these options to enable a simple transfer scheme for large resource handling, called blockwise transfer. While this approach works for non-lossy links, it uses a simple stop-and-wait strategy, which suffers major throughput problems in the presence of packet loss. To overcome this problem, our first publication [P1] introduces an efficient way to integrate reliable file transfer with network coding-based FEC into the protocol.

MQTT-SN: The Message Queue Telemetry Transport protocol for Sensor Networks (MQTT-SN) [46] is an adaptation of the popular MQTT protocol, which was initially developed for industrial condition monitoring, e.g., to remotely assess oil pipelines. The core of both MQTT variants is a publish-subscribe communication pattern, where a publisher sends its messages on a specific topic to a broker, which then forwards these publications to each subscriber of the topic. MQTT-SN is designed to enable this approach for constrained devices. While MQTT-SN features the core concepts of MQTT, it is designed to better work for constrained devices. Thus, it is adapted to wireless links with limited bandwidth, high packet loss rates, and small MTUs. The main difference to traditional MQTT to enable these advantages is the change of the underlying transport protocol. Instead of relying on TCP, MQTT-SN works on top of UDP. Any reliability mechanisms are implemented on the application layer in the protocol itself. This makes MQTT-SN favorable for the integration of packet-level FEC over traditional, TCP-based MQTT.

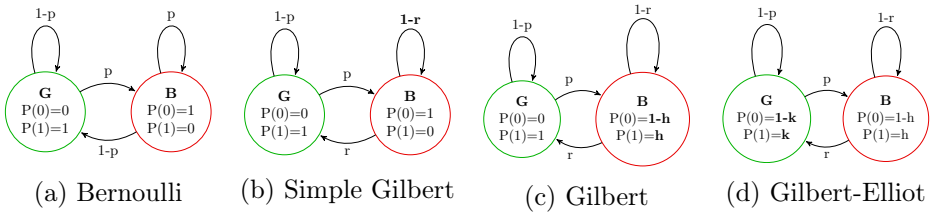
2.3 Evaluation

In our publications [P3] and [P5], we have introduced novel tools and methods to evaluate potential coding gains. Thus, the following section will summarize the relevant basics and present the related work, including an introduction to packet loss modeling and QoE for VoIP communication.

2.3.1 Packet Loss Modeling & Link Emulation

Reproducible packet loss modeling is necessary to enable a systematic evaluation of potential coding gains. Due to its prominence, this thesis will especially focus on the occurrence of packet loss in networks with a Ethernet-based 802.11 network stack, e.g., Wi-Fi. In such systems, packet loss mainly occurs either because of queue drops or due to bit flips on the medium. While queue drops are a result of active queue management (AQM) in the network interface controller (NIC) to prevent congestion and full buffers, these drops can hardly be reduced using packet-level FEC. Sending additional redundancy packets would stress the system even more, potentially leading to even more drops. Thus, we focus on the second case here, bit flip induced packet loss. Especially in wireless scenarios, this happens quite frequently due to signal degradation or interference on the physical layer. Yet, these physical effects are hard to model. Thus, often simpler loss models are used.

Because packet loss modeling is an essential part in all our publications, especially [P2], the following section highlights the benefits and drawbacks of different models. This is demonstrated using an exemplary packet loss distribution. A more formal introduction to packet loss modeling is given in [16]. The shown exemplary distribution is similar to the ones we have observed in the traces of a real-world WSN deployment [5]. There, most of the time only occasional packet losses happened, which may have been caused by interferences or signal degradation. Yet, sometimes, especially during periods of bad weather, the connection was heavily disturbed, which resulted in longer loss bursts. Figure 9a shows a loss sequence with these characteristics. Assume now, we want to train a model, which replicates this loss distribution. This is necessary to reproduce the observed scenario in a testbed environment when evaluating potential FEC gains. To highlight each models specific traits, we have implemented all here discussed models in *Sagemath* and rolled an exemplary loss sequence for each one. The models are shown in Figure 8 with their corresponding loss sequences given in Figure 9.



(a) Bernoulli (b) Simple Gilbert (c) Gilbert (d) Gilbert-Elliot

Figure 8: Packet Loss Models

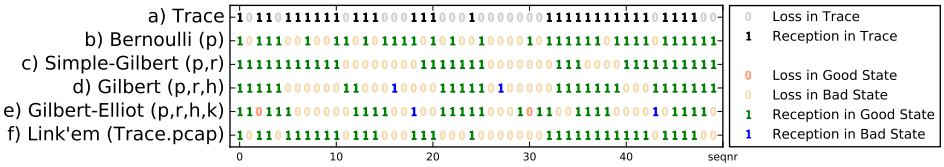


Figure 9: Loss models example.

Bernoulli Model: The simplest approach to model packet loss is to assume each packet has an independent loss probability. This is similar to conducting a weighted coin flip with probability of p to create a packet loss and probability $1 - p$ for a success. The resulting discrete binomial distribution can be created using a Bernoulli model, if we interpret output 0 as packet loss, and output 1 as a successful reception. Implementing a Bernoulli model can be done by using a two state Markov chain with fixed output states $G = 1$ and $B = 0$, and transition probabilities $P(G|B) = p$ and $P(G|B) = 1 - p$. Such a two state Markov chain is shown in Figure 8a. Exemplary rolling the model $n = 50$ times returns the packet loss sequence shown in Figure 9b. By comparing this sequence to the exemplary trace, the limitations of the Bernoulli model are obvious. Even though the model was initialized using the traces average PLR for p, the distributions differ heavily. While assuming a Bernoulli distribution allows for easy calculations, e.g. the average packet loss rate is simply p, real-world packet loss is seldom truly binomial distributed. Because the Bernoulli model does not have a form of memory, each roll is independent, and, thus, it is not possible to create longer loss bursts or extended sequences without losses.

Simple-Gilbert Model: This problem is fixed in the Simple-Gilbert model by using non-symmetric transition probabilities. So instead of using $P(G|B) = 1 - p$ for the transition back from the bad to the good state, a new parameter r is introduced, which leads to $P(G|B) = r$. This changes allows to

create bursty patterns. The corresponding Simple-Gilbert model is shown in Figure 8b. An exemplary loss sequence is presented in Figure 9b, which highlights the new burstiness, despite having the same average PLR. While this is a major improvement over the Bernoulli model, the Simple-Gilbert can not produce gaps within a burst sequence. The same output is created until a state transition happens. But even with this limitation, the model is still very useful to model loss sequences with two fixed states. For example, in our WSN deployments some sensor nodes usually worked perfect, but lost the connection to the base station completely while rebooting. During such interruptions, no packets at all could be received. This is a perfect case for the Simple-Gilbert model. Yet, our here shown exemplary trace does not have such distinct states. Even within a loss burst, some packets can still be received.

Gilbert Model: The more advanced Gilbert model addresses this issue by adding an additional output probability to the bad state, which is denoted by parameter h. When choosing $h > 0$, the model is able to produce receptions despite being in the bad state. These additional receptions are colored blue in Figure 9c. Comparing the Gilbert sequence to our exemplary trace shows a good similarity. Yet, there is still one open issue. The Gilbert model is not able to create occasional losses while in the good state.

Gilbert-Elliot Model: Finally, adding the loss probability $P(0) = 1 - k$ to the good state leads to the prominent Gilbert-Elliot model. Due to its ability to also create occasional errors without entering the bad state, the Gilbert-Elliot model has become state-of-the-art for bit error modeling. It has to be noted that all here mentioned Gilbert-based models were initially developed to model bit errors. Yet, nowadays they are also widely used to model packet losses, e.g., as implemented in the popular network emulated *netem*. The final Gilbert-Elliot model is shown in 8d with the new occasional packet losses are highlighted red in Figure 9d.

While the complexer approaches allow for more realistic packet loss modeling, this comes at the cost of additional parameters. The resulting greater parameter space has several disadvantages. First, it is harder to create formal equations. For example, formalizing the sRLNC coding gain (c.f., Equation 7) using a Gilbert-Elliot Model instead of a Bernoulli model is much more complicated. Secondly, the variance between sequences of the same model is higher due to the inherit burstiness. This is especially true if a model with low transition probabilities is used. As a result, a much greater

number of replications must be conducted to get significant testbed experiments. This combination of bloated parameter space and increased number of replications can lead to excessive experiment durations. Adding different codec parameterizations to the evaluation, e.g., changing g and r, amplifies the problem further. Thus, many network coding studies and also some publications of this thesis still emulate packet loss using a Bernoulli model to show general coding gains.

Trace-based Loss Emulation: While the aforementioned models allow to fit the loss distribution and average PLR of a collected trace, as shown in Figure 9, they rely on a random number generator (RNG), and, thus, cannot replicate the trace in a packet-by-packet form. For example, in our MQTT-SN work [P2], we trained a Gilbert-Elliot model to reproduce the packet loss characteristics of an observed, finished WSN deployment. Yet, this approach does not perfectly replicate the observed scenario. Doing so requires a trace-based loss models, as we have implemented using our *link'em* bridge [P3]. While the framework also allows to use all previously mentioned models, it is also possible to directly parse a *pcap*-trace into the emulator to exactly recreate a recorded loss pattern packet-by-packet. This trace-based emulation removes the variance completely to achieve more scenario-specific results at the cost of less generality. Such a trace-based loss sequence reproduction is shown in Figure 9f. More information on *link'em* will be presented in Section 3.2.1.

Link vs. Network Emulation: For this work, to evaluate the benefits of our FEC schemes, we only care about the link condition between encoding and decoding node. It is not necessary to recreate a complete network, but just emulate the link conditions between both interfaces, e.g., by applying the aforementioned models to emulate packet loss. Thus, this section focuses on link condition emulation technologies and not network emulators, e.g., MiniNet[13], or Core[2]. If one is interested in the research of network coding for larger topologies, there exists an integration of the popular network coding library Kodo into NS-3 [4], which enables scalable simulation. Also, this thesis aims to present the practical benefits of network coding for real-world scenarios. Thus, we believe that running real applications on real end-devices can achieve more meaningful results than relying on virtual testbeds. This said, some evaluations can only be conducted well in a real-world context. For example, for our MQTT-SN research [P2], we made precise energy consumption measurements of a Raspberry Pi Zero using a

[4] https://github.com/steinwurf/kodo-ns3-examples, accessed 02.08.2020

Fluke True-RMS digital multimeter. This would not have been possible with the same precision on virtual devices. With this strong believe in practical deployments, why should we still only emulate link conditions and not physically induce real packet loss? In short, physically recreating link conditions is a difficult task, which requires dedicated, expensive hardware. One notable approach to induce packet loss for network coding was conducted by Kim et al. by using a domestic microwave oven in close proximity to generate interferences [27]. Despite the ingenious idea and the experiments success, we gently doubt its long term viability and good reproducibility. Thus, we relied on the emulation of challenging link conditions in this thesis, especially to generate packet loss.

The arguably most prominent tool to reliably create artificial packet loss is the network emulator *NetEM* [5], which is part of Linux' Traffic Control (TC) packet. It has to be noted that *NetEM* can not only create packet loss, but also emulate delay, reordering, corruption and duplication. An empirical study of netems functionalities can be found in [24].

While *iptables* is usually one of the go-to tools for network administration, e.g. to increase the security of a network by adding packet filtering to incoming traffic, it can also emulate certain link conditions. Using the statistic mode of the *iptables-extensions* packet [6], it is possible to either drop random packets with a given probability, or even reject every n-th packet. Unfortunately, no complex models are implemented.

2.3.2 Quality of Experience for Voice over IP

Even tough traditional Quality of Service (QoS) metrics like delay, throughput, and packet loss are still the backbone of networking research, Quality of Experience (QoE) metrics have risen in popularity. These metrics try to quantify the quality of a system from an end-user perspective. The COST Action QUALINET defines the term Quality of Experience as follows [32]: *"The degree of delight or annoyance of the user of an application or service. It results from the fulfillment of his or her expectations with respect to the utility and / or enjoyment of the application or service in the light of the user's personality and current state."*

[5] https://man7.org/linux/man-pages/man8/tc-netem.8.html
[6] https://ipset.netfilter.org/iptables-extensions.man.html

QoE vs UX: It has to be noted that QoE is often associated or confused with User Experience (UX), which also focuses on the end users perception of a service or software. Even the COST Action definition of QoE is rather vague and lets room for interpretation. Thus, we stick to the tighter definition by Wechsung and De Moor [54]. They define QoE as primarily technology-driven and quantified with software-based, standardized measurement tools. UX on the other hand is characterized as a holistic approach with strong emphasis on qualitative research and done by interrogative user studies. With this in mind, this section focuses on QoE and not UX.

QoE Use Cases: One prominent example for the benefits of a QoE-centric evaluation is web browsing. Traditionally, Page Load Time (PLT) was the go-to metric to rate the performance of a web browsing server or hosting service. The PLT is calculated from the start of a request until the complete website is rendered. One drawback of the PLT is that it even includes elements, which are not in the current view port of the user. Recognizing this problem, the Speed Index (SI) metric uses a more user-centered approach by only considering the crucial above-the-fold (ATF) portion of the page. Thus, the SI better quantifies the real quality of experience. This change has aided in the development of new techniques like transmitting the ATF section of a website first using progressively rendering to achieve a higher user satisfaction.

Another popular research field for QoE is real-time video streaming. Because the impact of traditional QoS metrics like packet loss depends heavily on the video content, amount of movement between frames and cut frequency, these metrics cannot reliably represent the user satisfaction. Additionally, advanced compression techniques like inter-picture prediction, as used by the popular H.264 codec, have a significant impact on the rendered image. Thus, even traditional signal-based metrics like the Peak to Signal Noise Ratio (PSNR) are outdated and more complex QoE metrics are used to quantify the quality of such video streams. Popular examples are the Structural Similarity Index Measure (SSIM) or the Video Multimethod Assessment Fusion (VMAF). Later is used by Netflix to measure the impact of codec compression in their system.

QoE for VoIP: In our publication [P4], we have focused on QoE evaluation for VoIP applications for tactical communication in lossy networks. While running experiments with our *PyNC* network coding proxy, we realized that the existing QoE metrics do not cope well with packet loss. Traditional ap-

proaches, for example the Perceptual Evaluation of Speech Quality (PESQ) metric, were developed to evaluate small artifacts and distortions, which occur due to compression by the voice encoding. But especially in lossy scenarios, this minor speech signal degeneration is often overshadowed by packet loss-induced disruptions. For a good user experience, the amount of understandable words is much more important than the negative impact of small interferences, like noise or crackling. This is especially true for communication in tactical and command-and-control scenarios. Similar effects were not only shown for VoIP applications, but also in fundamental neurology studies regarding speech processing in the human brain, e.g., [37]. The following paragraph will present the most popular QoE metrics for VoIP, based on our related work section in [P5].

Perceptual Evaluation of Speech Quality (PESQ): The default metric for speech quality is the Perceptual Evaluation of Speech Quality (PESQ), which is standardized in ITU-T recommendation P862 [21]. PESQ was initially developed for telephony systems but was directly applied also for VoIP research. PESQ's basic algorithm aligns the reference and the recorded signals in time and uses a fast Fourier transform (FFT) for filtering. The raw PESQ value is then calculated based on disturbances in frequency and time, which are aggregated using an Lp norm. The conversion term to convert the raw PESQ values into Mean Opinion Score Listening Quality Objective (MOS-LQO) values is defined in ITU recommendation P.862.1 [22]. It has to be noted that the Perceptual Objective Listening Quality Analysis (POLQA, P.863, [23]) is the successor of PESQ and currently, the ITU recommended speech quality metric. The authors claim that POLQA *"offers an advanced level of benchmarking accuracy and adds significant new capabilities for super-wideband (HD) and full-band voice signals, along with support for most recent voice coding and VoIP/VoLTE transmission technologies"* [7] in comparison to PESQ. Unfortunately, there is no free POLQA implementation available to the public.

Hearing-Aid Speech Quality Index (HASQI): A non-ITU alternative is the Hearing-Aid Speech Quality Index (HASQI) [25], which was developed to measure the effects of distortions on the speech quality for persons in need of a hearing aid. HASQI is calculated by using two components. The first component measures how well the spectral representations of a signal fits its reference, measuring the effects of noise and nonlinear distortion. The second component considers the impact of linear filtering and spectral changes

[7] http://www.polqa.info/, accessed 04.08.2020

by analyzing the long-term average spectra. While there does not exist a public repository, a MATLAB code of HASQI Version 2 can be obtained by contacting the authors.

ViSQOL: Another notable VoIP metric is ViSQOL [18], which analyses spectro-temporal signal characteristics to model human perception. According to a study by the authors, ViSQOL achieves a closer correlation to the MoS than PESQ and POLQA. The software was recently made available to the public[8] but is still in early stages of development.

Further comprehensive information on the topic of QoE research in general can be found in [39]. The book also contains a summary of speech communication specific QoE, starting on page 165 in chapter twelve.

[8] https://github.com/google/visqol

2.4 Optimization

After the initial publication by Ahlswede et al. in 2000 [1], network coding has seen major research interest. While the idea was initially developed to solve routing bottlenecks, the concept of packet-level coding was extended to different scenarios. Especially the focus on its capabilities as a FEC scheme brought up new challenges and use cases, which required further optimizations. Because network coding stems from the field of information theory, some real-world implications were often neglected or simply unknown for a long time. For example, early works often used a non-systematic RLNC codec [33], where a linear combination is formed by all packets of the generation (c.f., Section 2.1). While this coding scheme can easily be implemented using matrix operations, it has a significant downside. Only if at least g packets are received, the decoded information can be forwarded, which results in large decoding delays. This issue can be fixed by using a systematic RLNC, where the first g packets are send basically uncoded, and, thus, can be directly forwarded to the receiver. The *TETRYS* codec [53] takes this early decoding one step further by using a sliding encoding window to make network coding-based FEC applicable to time critical applications.

Other optimizations focused on the computing complexity of network coding. For example, Silva et al. [41] presented a sparse coding scheme with overlapping generations, which reduces the complexity by allowing the decoder to alternate between Gaussian elimination and back substitution. A different approach was done by Lucani et al.[36], called Fulcrum network coding. This schemes enables a fluid allocation of coding complexity by nesting and levering different field sizes.

While the aforementioned optimizations mainly focused on the way in which the linear combinations are formed, another important real-world challenge is the handling of coding overhead. Because the optimization of this problem is the main focus of our contributions [P4] and [P6], the following section will explain the occurrence of coding overhead and summarize the existing approaches to reduce it.

2.4.1 Coding Overhead

The following section explains the occurrence of coding overhead based on our explanations in [P4] (c.f.[P4], Section II) and [P6] (c.f.[P6], Section II).

Header Overhead: To enable successful decoding, a coding header must be attached to each encoded packet, which contains at least the information about the used coding coefficients to form this linear combination. The size of this header depends on the generation size g and the used field size 2^q. Without further optimizations, this results in a header size of $g \cdot q$ bit. For $GF(2^8)$, which is the most popular field size, each coefficient represents one byte. Since g coefficients are used to form an encoded packet, the resulting minimal coding header has a size of g bytes ($g \cdot 8$ bits). Figure 10 visualizes the coding header overhead in bits for different generation sizes and field sizes. It has to be noted, that there are approaches to reduce the header size. One idea is to use a fixed codebook and only transmit the index of the encoded packet. If all parties share the same codebook, the coding coefficients can be looked up by their index within the codebook. Another approach is to transmit only a seed and initialize a random number generator, which then deterministically rolls the coding coefficients. As a drawback, such approaches either require additional information exchange or restrict the coding schemes. For example, recoding at intermediate nodes is a problem for seed-based coding. To form a fully functional protocol for a specific

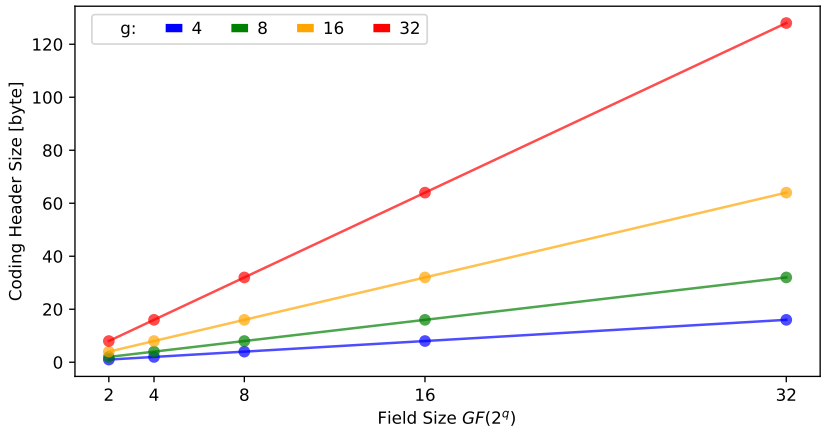


Figure 10: Coding header overhead in byte for different generation sizes and field sizes without optimizations (only coefficients).

purpose, the header should also include even more fields, e.g., a generation ID and the chosen generation size. An example for such a fully functional coding header is given in [10]. More information on coding headers and their optimizations can be found in the survey-style work of Heide et al. [17].

Padding Overhead: For this thesis, the header overhead is of marginal impact. Our studies [P4] and [P6] focus instead on a different type of overhead, called padding overhead, which is invoked by packets of heterogeneous lengths. Until the publication by Compta et al. [9] in 2015, it was often assumed that the only relevant coding overhead stems from the coding coefficients. This was based on the assumption that all original packets have the same or at least homogeneous packet lengths. Unfortunately, this is not the case for many real-world applications. While Compta et al. have used video streaming traces to highlight the problem, we presented heterogeneous packet length distributions in [P4] for other applications as well. Figure 11, which is taken from [P4], shows these in form of an empirical distribution function. The plot was created from collected traces of real-world applications and highlights the different levels of packet length heterogeneity. The UDP-based File Transfer Protocol (UFTP) for example creates packets of almost elusively the same length, as shown by the the magenta-colored graph. Skype VoIP on the other hand creates heterogeneous distributed packet sizes.

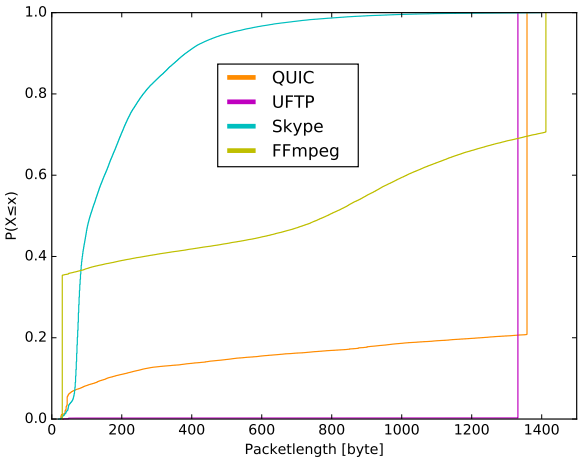


Figure 11: Empirical Distribution Function (ECDF) to highlight the packet length heterogeneity of collected real-world traces. (c.f., [P4], Figure 10).

To show why this packet length heterogeneity is important to consider, we must first explain how such a distribution leads to padding overhead. To do this, we start with traditional full-density RLNC example. Given a generation $G = \{p_1, p_2, p_3\}$ shall be encoded, which consists of $g = |G| = 3$ original packets. Assuming the original packet lengths are $|p_1| = 4$, $|p_2| = 2$ and $|p_3| = 3$ bytes. By using a field size of 2^8, one byte of data corresponds to one $GF(2^8)$-element, represented as one segment in the following Figure 12:

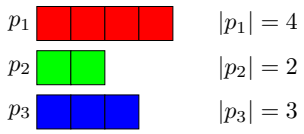


Figure 12: Original packets p_i with original lengths l_i

When using traditional full-density RLNC, an encoded packet q_i is created as a linear combination of all g original packets $p_i \in G$. Because we use $GF(2^8)$, each encoded byte is formed by combining the original bytes at the corresponding position in the packet. Therefore, all packets must have the same length before encoding. thus, all original packets p_i are padded with zeros to the length l_{max_G} of the longest packet in G, prior to starting the encoding process. When applying this method to our example, all packets will be pre-padded to length $l_{max_G} = |p_1| = 4$. This traditional, naive pre-padding approach is depicted by the gray blocks in figure 13:

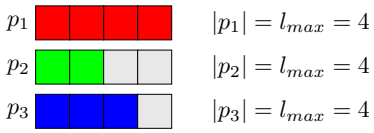


Figure 13: Original packets p_i padded to length l_{max_G}

The encoder then creates linear combinations of these pre-padded original packets. In our example, we want to create $r = 1$ additional redundancy packet. Therefore, the encoder hast to produce $g + r = 3 + 1$ encoded packets, namely q_1, q_2, q_3 and r_1. All resulting encoded packets have length l_{max_G}. The following figure 14 shows the encoding process for traditional full-density RLNC. The coloration represents which original packet influenced the corresponding encoded byte. Since the encoding process is based on byte-wise addition using the Galois-field arithmetics of $GF(2^8)$ a zero-padded byte has no influence on the creation of an encoded byte. In figure 14 the first two bytes of each encoded packet are influenced by all three original packets p_1,

p_2 and p_3, which is presented by the RGB-gradient. In contrast to this, each third byte of an encoded packet is only influenced by p_1 and p_2 (RB-gradient). And since p_2 and p_3 have zero-padding on the last byte, the value of the fourth byte of each encoded packet is only impacted by p_1 (solid red).

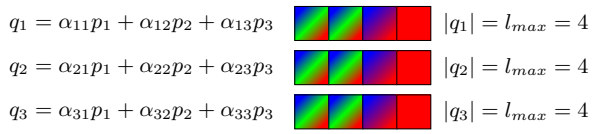


Figure 14: Traditional full-density RLNC with pre-padding

Using this full-density RLNC approach with pre-padding, each encoded packet has the length $l_{max} = |p_1| = 4$. As a result, the created encoded packets have a combined length of 16 bytes. Since the original amount of data is only 9 bytes, the total overhead of the shown RLNC encoding is 7 bytes.

For systematic RLNC this overhead does not occur for the first g packets, but only for the $g \cdot r$ redundancy packets, created in the second phase of the encoding algorithm. As shown in Figure 15.

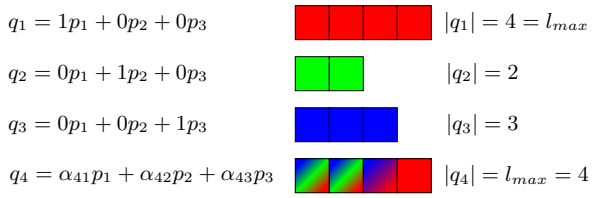


Figure 15: Padding overhead of systematic RLNC.

While this overhead seems marginal, its significance growth with increasing packet sizes. Thus, in our publications [P4] and [P6] we will evaluate the impact for the aforementioned real-world applications and show a new coding scheme, which reduces the overhead, while preserving the original packet structure.

2.4.2 Padding Reduction Schemes

In their initial work[8], Compta et al. did not only introduce the problem but also presented three simple approaches to solve the issue: fragmentation, bundling and chaining with fragmentation. The first algorithm, called simple fragmentation, cuts long packets into multiple small packets to achieve a better overall homogeneity. While simple fragmentation can reduce the padding overhead, sending more small packets can invoke additional overhead on lower layers. Unfortunately, the authors did only simulate the benefits of this scheme without a real network stack, and, thus, did not include the impact of this problem. The second strategy is named chop and bundle, which forms larger packets by concatenating multiple smaller ones. If very long packets exist, these are chopped into two smaller ones before concatenating them. This approach is very effective, but works only for packet distributions where multiple smaller fragments can be concatenated within the MTU limit. The third strategy, called chaining with fragmentation, concatenates the packets to a consecutive bytestream, then splits this chain into new, even-sized packets. Chaining with fragmentation showed the best results, reducing the padding overhead to below 5% (c.f. [8]). One drawback of all these strategies is the need to indicate how the new packets were formed, which requires additional information to be communicated. Secondly, the original packets have to be reconstructed after decoding, which can invoke additional processing delay. In [50], Taghouti et al. have analyzed additional traces to emphasize the significance of the zero-padding impact and verified the previous findings of Compta et al. [8].

A different approach, which is closer to our publication [P4], was presented by Taghouti et al. in [51]. Their solution divides the original packets into groups of bytes (if $GF(2^8)$ is used), called macro-symbols. Thereby, each packet is split into $n = \lceil \frac{l_{max}}{\mu} \rceil$ macro-symbols of length μ (cf. [**?**, sec. II.C]). The encoding is then done by combining subgenerations of macro-symbols. This approach is closer to our publication [P4] since it also uses the original packet lengths as information. While the macro-symbol approach uses a similar encoding scheme, it does not operate on a packet-level, and, thus, does not preserve the original packet structures.

3 Contributions

3.1 Integration

Our first two publications [P1] and [P2] introduce efficient ways to integrate intra-session network coding into application layer IoT protocols, namely the Constrained Applications Protocol (CoAP) and MQTT for Sensor Networks (MQTT-SN). The goal in both scenarios was to reduce the necessary radio uptime for reliable file transfer in lossy scenarios, e.g., when deploying over-the-air firmware updates in harsh weather conditions. Sticking to our motivational *bridging the gap* motto, we started building the network coding bridge from the practical side of things. The choice was made to experience possible real-world challenges on first hand and get our own insights into potential coding gains.

One thing, which distinguishes our approach from many related studies is the fact that we have truly implemented the coding schemes into both protocols and conducted the experiments in testbeds on real hardware. Because the network coding has its roots in information theory, this is not always the case. Many results are presented on a theoretical level or are verified only by simulation. Yet, we believe that running the code achieves more realistic results and can lead to new insights. This is in line with the unofficial IETF motto, defined by David D. Clark: *"We believe in rough consensus and running code"* [9].

[9] D.D. Clark, *"A Cloudy Crystal Ball: Visions of the Future"*, plenary presentation at 24th meeting of the Internet Engineering Task Force, Cambridge, Massachusetts, July 1992.

3.1.1 Adding a Network Coding Extension to CoAP for Large Resource Transfer

Extended Abstract (c.f.[P1]): Our first paper [P1] presents a smooth way to include network coding-based FEC in the Constrained Application Protocol (CoAP) for large resource transmissions. IoT devices usually communicate using short messages with little data. In some cases, for example, requesting firmware updates, bigger resources need to be transferred. While CoAP's normal blockwise transfer scheme can handle large resources, it is not efficient in lossy environments. The paper first demonstrates the limitations of CoAP's existing blockwise transfer scheme, then presents a new approach to overcome this problem by using network coding-based packet-level FEC. Our proposed extension introduces two new CoAP message options, called *NC Control* and *NC Coefficients*. These are used to implement a three stage file transfer scheme, consisting of an initialization, a transmission, and a termination phase. In conjunction with the new message options, the scheme was implemented into the *cantcoap* library [10] in form of an additional extension layer, as shown in Figure 16. Testbed measurements using an implemented client-server application with emulated packet loss and delay confirm the advantage of our network coding extension over CoAP's regular blockwise transfer. The resulting signifiifantly reduced file transfer durations are highlighted in Figure 17 for different codec parameterizations and block sizes.

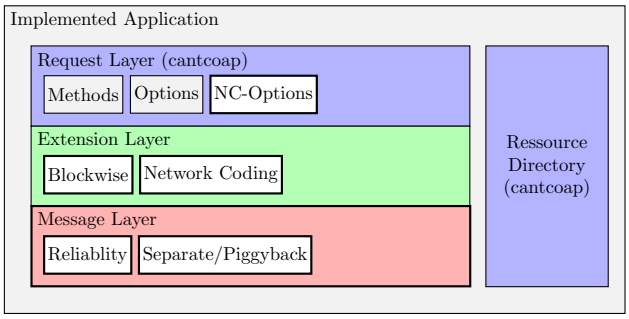


Figure 16: Application structure of the implemented network coding-based transfer for CoAP (c.f., [P1], Figure 9).

[10] https://github.com/staropram/cantcoap, accessed 18.11.2020

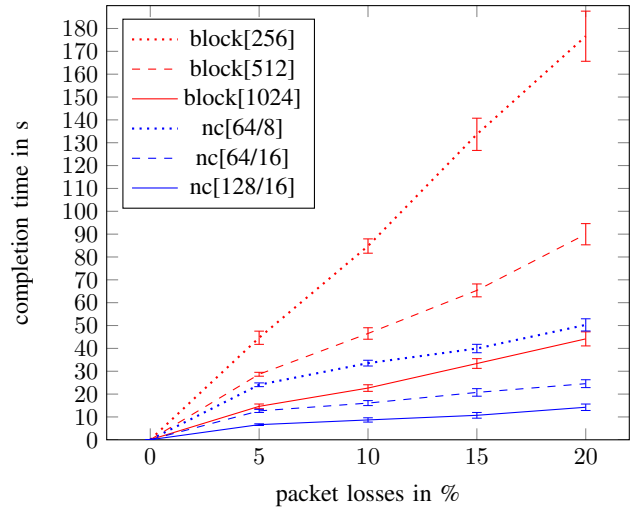


Figure 17: Comparison between CoAPs traditional blockwise transfer and the novel network coding extension (c.f., [P1], Figure 14).

Context & Discussion: As shown in the road map for this thesis, c.f. Figure 2, this study was the start of our network coding research. The motivation for this work was to get first hand experience with network coding, see potential coding gains in practice and learn from the given challenges. The thereby learned lessons and findings were very helpful in the development of the integration extension for MQTT-SN [P2], as will be further discussed in section 3.1.2.

The main challenge was the development of a stoppage strategy without the usage of much feedback that operates within the bounds of the underlying protocol. For the targeted use case of over-the-air firmware updates, an early stoppage is especially problematic. Sending an insufficient number of redundancy packets results in non-decodable generations, which can lead to massive data inconsistency and even non-functioning devices (bricks). Yet, sending excessive packets clogs the link and wastes valuable bandwidth and radio uptime. Thus, the total file transmission system still requires small amounts of feedback to inform the sender about successful decoding. To solve this issue, the introduced CoAP extension uses a three-phase transfer scheme. After initializing the transfer and transmitting most data, the final termination phase is started by the server, when encoding the last generation of the requested resource's data. This is communicated from the server to the client by using a flag in the responses of our novel introduced

NC_ Coefficients option. After decoding the last generation, the client returns a confirmable message to the server, including a *NC_ Control* option with the same flag also set. When receiving this stop-message from the client, the server acknowledges it and stops creating coded packets. The transfer is then reliably completed. While this approach does not only solve the stoppage problem with minimal use of feedback, it is also in line with the usage of the *more* flag in CoAP's traditional blockwise transfer.

In regards to related work, the developed extension for CoAP [P1] (and also for MQTT-SN [P2]) can be closest compared to the FEC extension for QUIC [42]. In both cases, an existing transport/application layer protocol is enhanced with coding to supply a reliable communication architecture for the final application itself. It has to be noted that our work was published first. Paper [P1] was presented in November 2016 and [P2] was presented in October 2017. The first draft of the QUIC extension by contrast was proposed later in March 2018 [42]. While all extensions level network coding-based FEC to overcome packet loss, the underlying codec differs. Our studies use a seed-based RLNC codec with fixed generations, the QUIC extension instead implements a sliding window coding scheme, which is similar to the TETRYS codec [53]. Despite not reducing the total transfer duration, using such a sliding window codec can reduce the decoding delay by making individual packets earlier available to the decoder. In our use case of firmware updates, this is not really important, but for applications with stronger real-time constrains, e.g., web browsing via QUIC, it can be beneficial. One drawback of a sliding window codec is the need for feedback to inform the encoder about the optimal window size to enable reliability. Our system design instead guarantees reliability with the aforementioned stoppage algorithm in conjunction with a more simple, seed-based RLNC codec that requires less feedback. Yet, the developed architecture still allows to switch the underlying FEC codec, which is an interesting topic for future work.

3.1.2 Improving Energy Efficiency of MQTT-SN in Lossy Environments Using Seed-Based Network Coding

Extended Abstract: This paper presents an energy-efficient solution to overcome packet loss in Wireless Sensor Networks (WSNs) by adding seed-based Network Coding to MQTT for Sensor Networks (MQTT-SN). Whereas most sensors in common WSN devices consume little energy, using the radio is costly. Thus, devices try to minimize their radio uptime, while still satisfy timeliness and reliability of data delivery. The proposed approach uses an optimized seed-based Network Coding scheme for Forward Error Correction to shorten the sensor node's radio uptime and reducing its power consumption. The solution is conform to the MQTT-SN specification and, thus, interoperable with existing systems. The presented evaluation is based on collected traces from a real-world WSN deployment in the context of Precision Agriculture. Radio uptime and power consumption measurements in an experimental testbed confirm the achieved benefits. The potential daily energy savings are highlighted in Figure 18, if the proposed coding approach is applied to the links of the motivational WSN deployment. As indicated by the same-colored dotted and solid lines, a saving of up to 127.25 mWh (43.79 %) per day (Link $B \rightarrow G$) is possible, depending on the link quality. On average, the network coding extension reduces the consumed energy in the transmission phases by 96.70 mWh (38.21 %) in our motivational scenario.

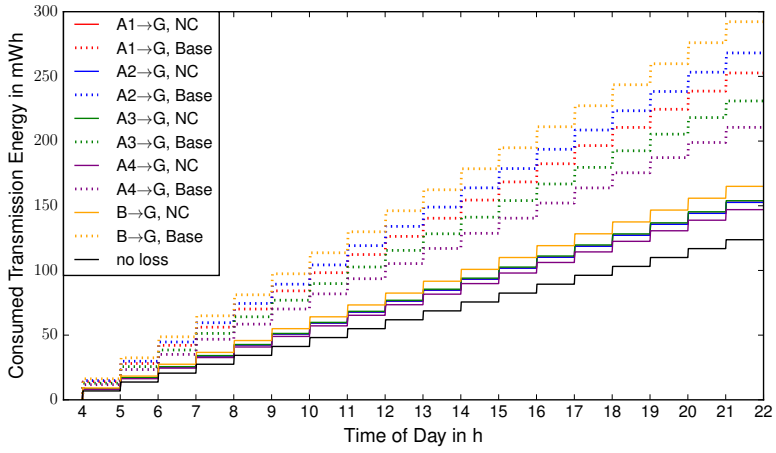


Figure 18: Predicted daily radio interface energy consumption for each link of the motivational WSN deployment(c.f., [P2], Figure 7), if network coding is used (NC) in comparison to the basic approach (Base).

Context & Discussion: The initial motivation to explore potential coding gains for MQTT-SN was given by the challenges, which were observed in a WSN deployment of our college Jan Bauer. Due to a combination of poor weather conditions and constrained transmission power of the used Telos B sensor nodes, the system suffered from heavy packet loss. Our previously conducted CoAP study [P1] had already verified the potential of an FEC extension to overcome such a problem and reduce the transmission duration for large file transfers. Because a good network trace data base was collected during the deployment, a unique opportunity was given to reproduce the deployment in our lab and explore the benefits of network coding in a more practical scenario.

While it would have been possible to choose other integration approaches or protocols, MQTT-SN can especially benefit from a network coding extension, because it implements the Iot-friendly publish-subscribe-based communication pattern on top of UDP. As mentioned before in Section 2.2.1, network coding integration favors UDP over TCP as the underlying transport protocol. The main problem of TCP-based approaches, such as traditional MQTT, lies within the underlying automatic repeat request (ARQ) retransmission scheme of TCP, which interprets packet loss as congestion. TCP was initially developed for wired communication, medium-induced packet loss was simply

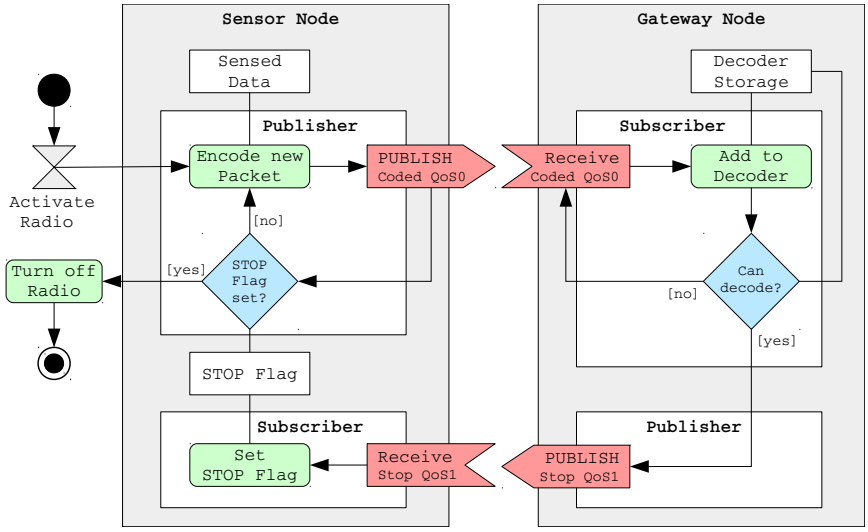


Figure 19: Activity diagram of one transmission phase with our MQTT-SN network coding extension (c.f., [P2], Figure 3).

not thought of. This problem can cause a large amount of retransmissions or even lead to connection timeouts, as highlighted in the practical study by Bauer et al. [6]. In such scenarios, our developed approach achieves a higher reliability, not only because MQTT-SN is based on connectionless UDP, but also because the network coding-based FEC extension adds an additional layer of robustness.

While there are similarities between our MQTT-SN extension [P2] and our CoAP extension [P1], especially MQTT-SN's underlying publish-subscribe communication pattern offered new challenges. Our final solution to integrate network coding-based FEC into MQTT-SN's publish-subscribe architecture is shown in Figure 19. Instead of transmitting all queued data as *QoS 1* messages and wait for acknowledgements, the publisher sends with *QoS 0* until a subscriber has received enough encoded packets to restore all original data by decoding. This guarantees reliable transfer without possibly suffering from retransmission delays. If the decoding-condition is satisfied, the decoder sends out a *STOP*-control message to inform the encoder. This single control message is send as a reliable *QoS 1* publication, using MQTT-SN's build-in ARQ mechanism. When the subscriber at the sensor node receives this control message, an internal flag is activated, stopping the publication of coded packets. Again, similar to our CoAP extension [P1], this approach does not work completely without feedback, but keeps the necessary amount to a minimum, while still guaranteeing reliability. Because the sender does not have to wait for acknowledgements or retransmit lost packets, file transfers can be completed faster. Thus, the sender can turn off its radio and enter its deep sleep phase earlier, which leads to the energy consumption savings shown in Figure 18.

Because we did implement our coding scheme, it would have been possible to follow up the testbed studies with a real-world deployment. While [P1] used a simple Bernoulli packet loss model, the evaluation in [P2] was already based on collected traces. Observing a real-world deployment would have been the next logical step and could have yielded very interesting results. Unfortunately, no opportunity for such a deployment occurred. Thus, we shifted our focus to the research of novel evaluation tools and methods to better quantifiy possible network coding gains, which will be presented in the upcoming section. Especially the developement of our link emulation bridge *link' em* was heavily influenced by the work on [P2].

3.2 Evaluation

Sticking to our *bridging the gap* motto, we move on to the second pillar of our network coding bridge, which is the development of new methods for a better evaluation of potential network coding gains. The following section highlights and discusses our contributions to the field. These are namely the development of a novel layer-2 bridge for reproducible link emulation, called *link' em* [P3], and a machine learning-based QoE metric for VoIP traffic, called *SPQER* [P4].

3.2.1 Link 'em: An Open Source Link Emulation Bridge for Reproducible Networking Research

Extended Abstract: Publication [P3] presents *link 'em*, an open source link emulation bridge for reproducible networking research. While reproducibility is one keystone of good research, most available link emulators are lacking crucial features or are prohibitively expensive. *link 'em* is essentially a Raspberry Pi-based layer-2 bridge that runs an extended version of *netem* in conjunction with a trace database to achieve reproducible link emulation. Because the node has two Ethernet interfaces, it can be used to connect two hosts and emulate the link conditions between them, without any configuration on the end devices them selfs. An image of a *link 'em* prototype with a custom 3D-printed casing is shown in Figure 20. The *link 'em* framework also contains a novel packet loss module, which builds upon *sagemath* and *NetfilterQueue*, to run more sophisticated packet loss models in addition to *netem's* existing ones. An overview of the general system architecture is presented in Figure 21.

Figure 20: *link 'em* prototype in a custom 3D-printed case.

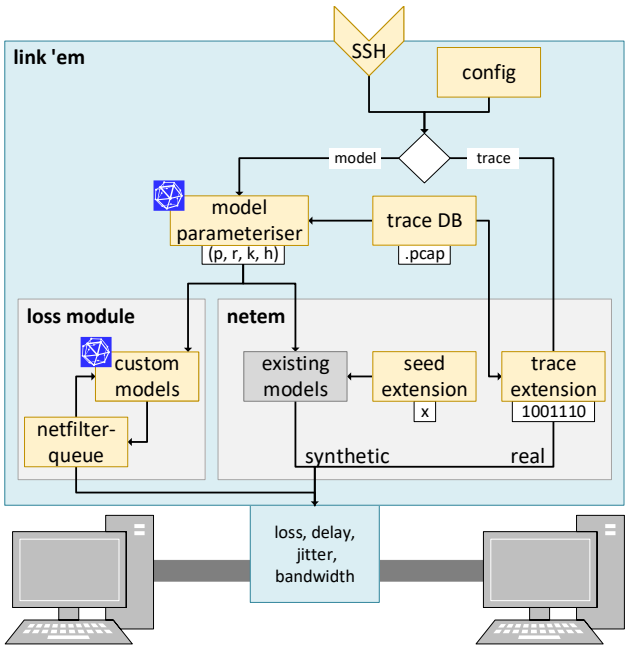


Figure 21: *link 'em* system architecture (c.f., [P3]).

Context & Discussion: While inter-session network coding is theoretically beneficial even without the presence of packet loss, e.g., by solving routing bottlenecks, intra-session FEC is only useful if there are packets to recover. Thus, the evaluation of potential FEC benefits requires a proper packet loss emulation. Yet, emulating a link nad reliably reproducing packet loss distributions is easier said than done. We stumbled upon this challenge during our integration of network coding into MQTT-SN [P2]. During the deployment of an in-situ WSN [5] to measure the growth of crop, the gateway node also recorded network traces for further analysis. Upon inspection, these traces contained vast amounts of packet loss. Depending on the sending node, between 13.38% and 19.42% of all transmissions failed. Additionally, these losses did not follow a Bernoulli distribution but occurred in heavy bursts. Following the state-of-the art, a Gilbert-Elliot model was parameterized to reproduce the link conditions in the lab for our MQTT-SN testbed. Unfortunately, due to the burstiness, the trained model had one good and one very bad state, with small state transition probabilities. This resulted in strongly fluctuating outcomes, depending on how the transitions were drawn by *Netem's* internal random number generator. While this was sufficient for a general evaluation of our MQTT-SN FEC extension [P2], the model

Figure 22: *link 'em* demonstration testbed for reproducible packet loss emulation (c.f., [P3]).

just emulated a similar loss distribution but could not exactly replicate the observed loss pattern of the deployment. It would have been very interesting to see how our approach had fare in direct comparison to the real-world deployment, if it was tested packet-by-packet against exactly the same loss pattern. Such a trace-based loss emulation is not possible with traditional loss emulators, but can now be done using our *link'em* bridge.

An early prototype of the bridge was shown at the NetSys 2019, where is was received very well by the scientific community and members of the industry alike. A refined version was then successfully presented at the LCN demonstration track, where it won the best demonstration award. Figure 22 shows the testbed in our lab, which is similar to the one used in the LCN demonstration. The showcase mainly highlighted the trace-based packet loss emulation feature of *link'em*. This was done by showing the impact of a model-based link emulation via *netem* and a trace-based emulation using our *link 'em* bridge on a video live stream, in comparison to a pre-captured reference stream. As seen in Figure 22, the left monitor (black) displays the original pre-recorded video with packet-loss induced artifacts. While the *link'em*-impaired stream (yellow) shows similar artifacts to the captured video, the current frame of the *netem* impaired video looks different. In the content of this thesis, *link'em* was also used in the *SPQER* [P5] testbed to evaluate the impact of packet loss on VoIP QoE.

For the LCN demonstration track submission, we created an extended ver-

sion of our initial paper [P3]. Because it was not published in the main conference proceedings, we uploaded the extended version to our project's website [11]. A more detailed description of the setup and the necessary steps to install and deploy the bridge can also be found there. For the LCN demonstration, we also improved the software and fixed minor issues. These improvements from *link'em v1* (shown at NetSys) to *link'em v2* (shown at LCN) are visualized in Figure 23. While the first version already worked better than traditional *Netem*, it still had had some inconsistencies in terms of reproducibility. Although the packet loss was replayed on a packet-by-packet basis, the reproduced video still showed minor differences to the original reference video. This lead to a reduced SSIM score, which can be seen in the left subplot of Figure 23. The visible dips of the golden *link'em v1* graph were the result of an odd queuing behavior in the internal delay buffer of the underlying Netem implementation. Yet, even with this issue, *link'em v1* (golden graph) achieved a better score than traditional *Netem* (blue graph). After fixing the queuing problem, the second iteration of our software, *link'em v2*, achieves a nearly perfect SSIM score, as highlighted by the green graph. While each line in the left subplot only visualizes the SSIM scores of the first 2500 frames, each boxplot on the right side contains the data for all transmitted 11093 frames. It has to be mentioned, that we did not re-evaluate our experiments with these changes merged backed into the traditional *netem*, which is still noted future work.

One point of critique, which was brought up at the LCN demonstration, is the choice of a Raspberry Pi as the underlying hardware platform. Compared to full-sized desktop systems, the Pi's performance is rather limited, which can restrict the potential application of *Link'em*. For example, while the emulation of packet loss to evaluate the performance of TCP connections in high speed gigabit WiFi yields very interesting research questions, the on-board Ethernet port and potentially also the CPU of the Raspberry Pi is simply to slow for this use case. However, the *link'em* framework also runs on most Linux-based systems and can easily be installed on different platforms. Thus, a computer with better specs can be used to overcome possible performance problems. We have already verified this approach by building a multi-path prototype on the basis of a Dell Optiplex 5070 desktop computer. This new prototype is equipped with two DeLock i350 four-port PCIe network cards, which enables the emulation of packet loss and delay for multi-path scenarios with upto four links.

[11] https://sys.cs.uos.de/linkem/index.shtml

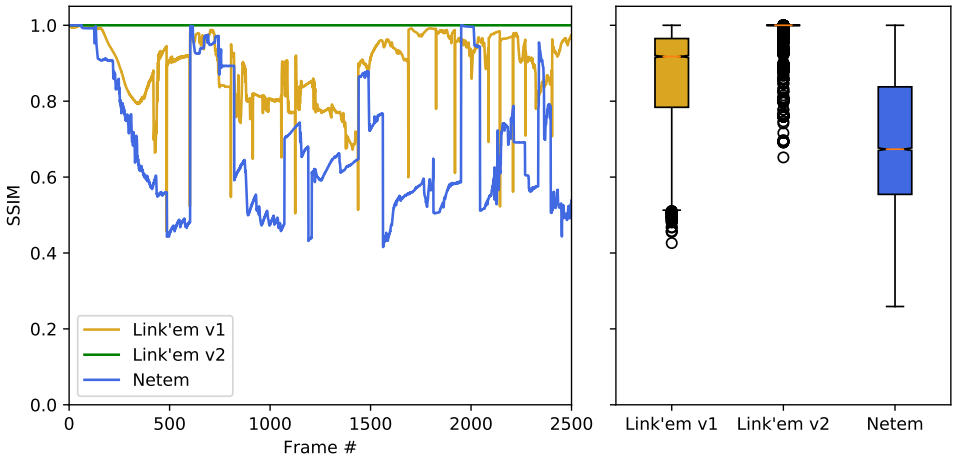


Figure 23: Improvement from *link'em v1* [P3] to *link'em v2* (LCN Demonstration, https://sys.cs.uos.de/linkem/index.shtml) for the SSIM score of a transmitted video.

While *link'em* was initially developed to enable trace-based replay of packet loss sequences, its integrated custom loss module can be also used for other purposes. Because the loss module uses the *NetfilterQueue* library to link a socket to an *iptable* rule set, it is possible to fetch, alter or drop specific packets. For example, our co-author Dominic Laniewski has used this approach in his publication *"On the Impact of Burst Loss for QoE-Based Performance Evaluations for Video Streaming"* [30]. There, the custom loss module of *link'em* is extended to drop specific packets within an RTP video stream. The recorded video is then processed with QoE metrics to evaluate the impact of specific loss sequences. This study does not only highlight the versatility of *link'em* but also gives a glimpse on its potential for future research.

3.2.2 SPQER: Speech Quality Evaluation Using Word Recognition for VoIP Communication in Lossy and Mobile Networks

Extended Abstract: Publication [P5] introduces SPQER (pronounced speaker), a novel approach to evaluate the quality of experience for real-time Voice over IP (VoIP) communication in mobile and lossy networks. Since most VoIP applications aim to provide low latency communication, they are based on RTP over UDP. Thus, there is no transport layer reliability to recover lost packets. This can lead to gaps in the rendered speech signal at the receiver, resulting in non-understandable words, which ultimately reduces the user experience of the VoIP call. Figure 24 depicts such a problem. While the error-free transmission (blue) does not contain loss-induced gaps, the error-prone recording (yellow) is heavily interrupted. Traditional speech quality metrics, e.g., Perceptual Evaluation of Speech Quality (PESQ) or the Hearing-Aid Speech Quality Index (HASQI), can not cope well with such heavy interruptions, because they directly compare frequencies and amplitudes to calculate the received signal distortions. SPQER instead uses machine learning classification to evaluate the percentage of recognizable words in conjunction with a time-based decay function to penalize delay and cross-talking. So instead of evaluating noise, SPQER directly answers the question: What percentage of words is the recipient able to understand? The paper includes a sensitivity analysis, which is based on testbed experiments for different packet loss rates and simulated delays, to asses the impact of challenging link conditions. As shown in Figure 25, a final correlation analysis to a short user study shows that SPQER can better evaluate the amount

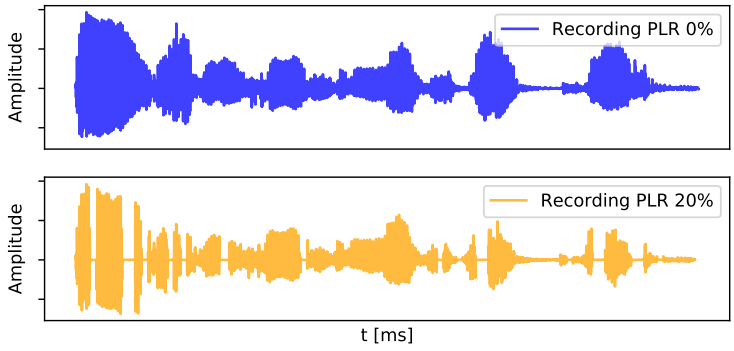


Figure 24: Exemplary impact of packet loss on VoIP audio recordings.

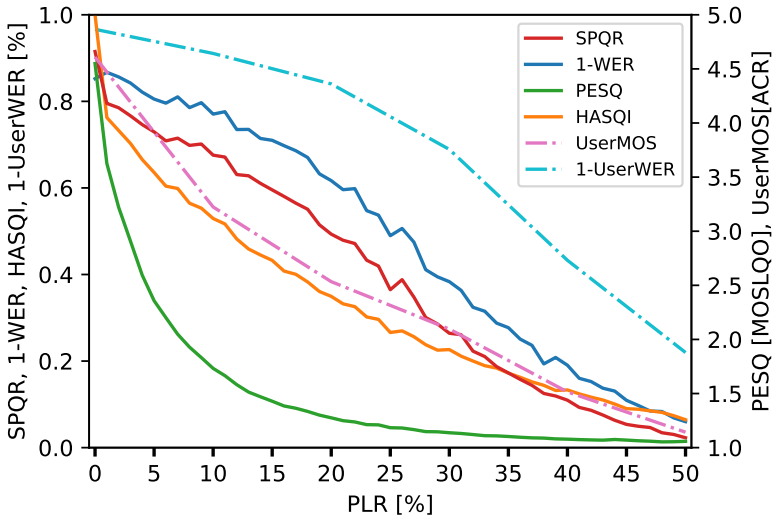


Figure 25: Correlation of SPQER and other VoIP metrics to the results of a short user study for different packet loss rates (c.f. [P5]).

of understandable words than PESQ and HASQI, while still giving a more precise indication about the voice quality than the Word Error Rate (WER) metric.

Context & Discussion: Most network coding research focuses on the impact of network coding in terms of packet loss reduction. While this QoS perspective is reasonable, it does not directly allow to make assumptions about the coding gains in terms of end user experience. For example, when streaming a video, even little amounts of packet loss can lead to heavy artifacts, as shown in our co-authored work [30]. Other applications are more robust and can cope better with packet loss, for example web browsing. Thus, it is important to use application-specific QoE metrics to truly quantify the potential benefit of network coding for the end user.

The motivation to develop *SPQER* was given by the challenges, which we observed while conducting QoE testbed experiments for a variety of application types to find optimized, application-specific codec parameterizations.The therefore build testbed consisted of two Dell Optiplex desktop computers, which were connected by a *link'em* bridge [P3] to emulate packet loss and delay. This setup once more showed the usefulness of *link'em* for reproducible networking research. Upon listening to exemplary collected loss-prone VoIP recordings, we realized that many words were not understandable

due to gaps in the rendered speech signal. This problem is highlighted in Figure 24. Thus, the following research question arose: *How can we quantify the amount of understandable words and by how much does network coding-based FEC increase this value?* While the existing speech quality metrics, e.g., PESQ [21] and HASQI [25] are made to evaluate disturbances, they do not directly answer the question. This led to the development of SPQER, which was finally published in the IEEE Open Journal of the Computer Society in July 2020.

While SPQER was developed to quantify potential coding gains, publication [P5] only introduces the SPQER metric and does not include an evaluation for network coding as it would have stretched the scope of the paper. Thus, to complete the picture, we will present a coding gain approximation for VoIP in terms of SPQER QoE here. Unfortunately, due to parameter space scaling, it is not possible to replicate the testbed experiments of the SPQER journal publication for all relevant codec parameterizations. The original experiments were conducted in physical testbed with the audio recording running in real time. This took almost six weeks for the data set of [P5]. Running the setup for different generation sizes and redundancy rates would blow up the parameter space, making this approach infeasible. Thus, instead of replicating the testbed experiments, we approximate the corresponding SPQER value for each codec parameterization by using the corresponding remaining packet loss rate after decoding (c.f., Figure 5). For example, as previously shown in Figure 5, applying systematic RLNC with $g = 8$ and $r = 0.5$ on a link with $PLR_L = 25\%$ results in a final loss rate after decoding of $PLR_{DEC} = 7.2\%$. The *SPQER* value for this parameterization can then be approximated by taking the the one of the best fitting link loss rate of the [P5] experiments. Thus, for the aforementioned example parameterization, we assume:

$$SPQER_{NC}(PLR_{DEC} = 7.2\%) \approx SPQER_{[P5]}(PLR_L = 7\%).$$

While this approach does not yield perfectly accurate results, it still allows to make reasonable assumption about possible coding gains. The so approximated results for different SRLNC parameterizations are presented in Figure 26, where the gray graph represents the *SPQER* values from [P5] as the uncoded reference. As shown, network coding-based FEC can greatly increase the QoE of VoIP applications. For our previous example ($g = 8$, $r = 0.5$, $PLR_L = 25\%$), the *SPQER* value can be nearly doubled from 0.36 to 0.73. This can be seen in Figure 26 by comparing the green crosses at

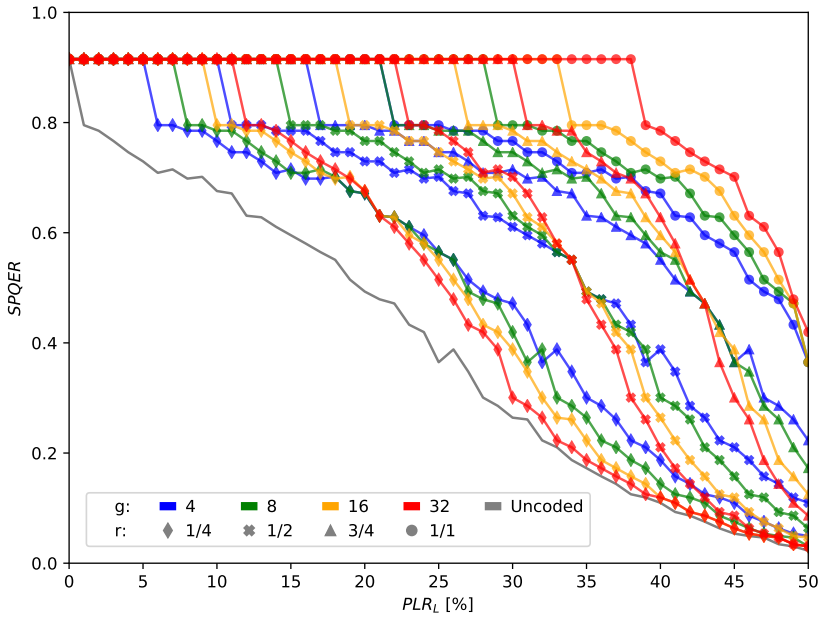


Figure 26: Approximated impact of network coding-based FEC on *SPQER* for the evaluation of VoIP applications.

PLR_L = 25% to the gray-colored uncoded reference. Other parameterizations, especially those with higher redundancy rates, lead to an even larger gain. For example, when using $g = 32$ and $r = 1.0$, the QoE is still perfect for packet loss rates of upto 35%.

Regarding the publication itself, one justified point of critique, which was mentioned by one reviewer of the journal submission, is the usage of a general-purpose classifier. The results in [P5] were calculated using the *Google S2T* classifier[12], which was trained on carefully created annotations of known texts to achieve the highest score in correspondence to this ground truth. Yet, such a classifier does not directly reflect the true human hearing capability. Thus, it would be interesting to train a special classifier for *SPQER*, which does maybe not achieve a perfect score compared to the annotation data set, but instead better mimics the human reception of speech.

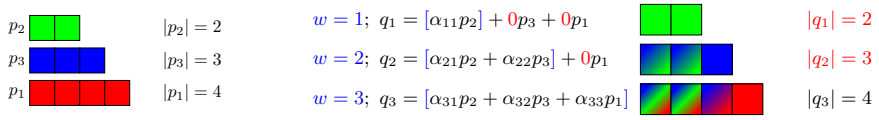
[12]https://cloud.google.com/speech-to-text, accessed 03.03.2021

3.3 Optimization

The third and final set of contributions is the evaluation of coding overhead and the introduction of an optimized coding strategy for applications with heterogeneous packet lengths. Because network coding stems from the field of information theory, some real-world implications were often neglected. As mention in Section 2.4.1, one such topic is the occurrence of coding overhead. Here, we focus especially on the padding overhead, which is invoked by packets of heterogeneous lengths.

3.3.1 Packet-Preserving Network Coding Schemes for Padding Overhead Reduction

Extended Abstract: While many network coding studies assume homogeneous packet lengths, this is often not the case for real-world applications (c.f., Section 2.4.1, Figure 11). Such heterogeneity results in excessive padding overhead and is a critical problem when using network coding for real-world data in the wild. Current coding schemes either apply excessive zero-padding or rely on communication-intensive packet reconstruction strategies, i.e. bundling or chaining. Our contribution [P4] presents novel encoding strategies to reduce the padding-induced overhead while preserving the original packet structures. The most sophisticated one, called size-based coding with padding-on-demand, is shown in Figure 27. Instead of padding shorter packets prior to encoding, the original packets are first ordered ascending to their size and then encoded using a sparse, growing coding window. Only packets within this coding window are combined during the encoding process. Thus, shorter packets must only be padded to the lengths of the longest on in this window. The conducted evaluation is based on traces of datagram-based real-world applications, namely browsing via QUIC, Skype



(a) Original packets sorted by size.

(b) Encoded packets using a sparse, size-based coding window.

Figure 27: Size-based Coding with padding-on-demand for padding overhead reduction (c.f. [P4], Figures 8+9).

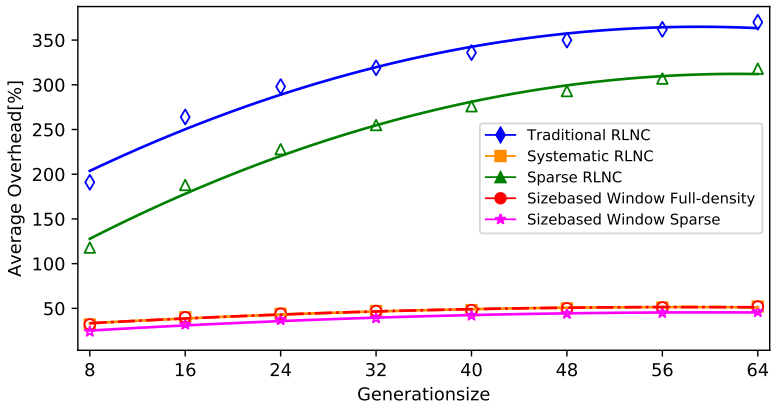


Figure 28: Average coding overhead of size-based coding in comparison to existing coding schemes. The results are shown for the Skype trace using a fixed redundancy rate of 12.5%. (c.f. [P4], Figure 12).

VoIP, FFmpeg video streaming, and UFTP file transfer. The results verify that the packet-preserving coding schemes can significantly reduce the overhead of transmitted bytes for different generation sizes and packet loss rates without the need for additional communication about packet reconstruction. This is exemplary highlighted in Figure 28 for the Skype trace.

Context & Discussion: We first observed the problem of excessive padding overhead while developing a transparent Python-based network coding proxy, called PyNC. During testing with the traditional full-density RLNC codec [19], a huge discrepancy was detected between the amount of original uncoded payload and the number of transmitted bytes. This overhead is shown in Figure 29. The existing solutions to reduce such overhead, e.g., fragmentation, bundling and chaining by Compta et al. [8], require the transfer of additional information about the recomposition process. This can be problematic in our proxy use case, for example, if MTU restrictions apply. To solve this issue, the goal of our research was to develop a novel coding scheme, which allows padding reduction without invoking additional communication overhead. The result was the aforementioned size-based coding scheme, published in [P4].

As shown in Figure 29, the possible gain of padding overhead optimization depends heavily on the applications packet length distribution. The more heterogeneous the packet lengths are, the more padding occurs. Thus, applying a coding overhead optimization is only really necessary and beneficial

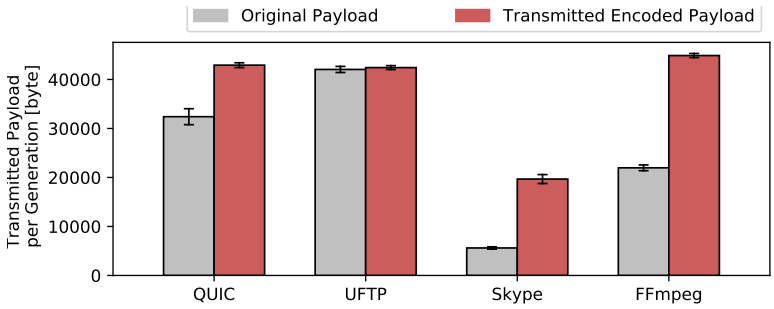


Figure 29: Observed average transmitted payload per RLNC generation (g=32) in the PyNC proxy for UDP-based applications on an error-free link. The error bars show 95% confidence intervals (c.f. [P4], Figure 2).

for applications with heterogeneous packet lengths. For example, our integration solutions [P1] and [P2] were developed to speed up over-the-air firmware updates. In such a scenario, all data is available a priori and can be fragmented into even-sized chunks. The resulting packet length distribution is similar to the one of a UFTP file transfer, where almost no padding overhead occurs (c.f. Figure 29). Thus, it is not really necessary to apply any padding optimization. The possible gain is much higher for real-time multimedia application like Skype VoIP, where the data is generated during runtime and must be transmitted with tight timeliness constraints. An overhead evaluation for different application types was done in [P4], which verified this hypothesis.

It has to be noted that a related coding scheme was proposed by Taghouti et. al in their publication *"Reduction of Padding Overhead for RLNC Media Distribution With Variable Size Packets"* [52]. While there are similarities to our work, [P4] was independently developed and earlier submitted for publication. We initially submitted [P4] to the Conext conference in June 2018, where it got just about rejected after a 3 month long review process. The paper was then submitted to the IEEE Conference on Local Computer Networks (LCN) in March 2019, where it was accepted and published in the proceedings via IEEE Xplore. We later discovered the paper by Taghouti et. al in the IEEE Transactions on Broadcasting journal, which was published in September 2019. While both solutions have a similar motivation and show a trace-based evaluation to highlight the impact for real-world applications, there are distinct differences in the encoding process itself. Our size-based coding scheme combines the original packets and preserves their structures,

and, thus, can be applied to all types of traffic. The approach by Taghouti et. al instead focuses especially on video traffic and introduce a new notion, called macro-symbols. These macro-symbols can correspond to a single GF-element or to a series of elements within one original packet. The padding overhead reduction is then achieved by cleverly selecting and ordering the macro-symbols of a generation prior to encoding, Yet, this scheme does not directly preserve the original packet structures and requires additional information about the macro-symbol forming, which is the main difference to our approach.

3.3.2 The Impact of Bit Errors on Intra-Session Network Coding with Heterogeneous Packet Length

Extended Abstract: Recent studies, e.g., [8] and our previous work [P4], have shown that network coding-based FEC can invoke increased packet sizes, and, thus, excessive padding overhead. This problem is especially significant for traffic with heterogeneous packet lengths. Yet, most conducted studies, including our previous ones, emulate packet loss using length independent random loss models. In reality, larger packets have a higher probability to get corrupted, e.g., by failing a series of link layer cycling redundancy checks. Thus, additional padding can not only have a negative impact on the throughput, but also possibly increase the overall frame loss rate and the packet loss rate. To highlight this problem, Figure 30 exemplary shows the increased average frame error rate, which is induced by the additional coding overhead, if a bit error model is applied. To further analyze the magnitude of this problem, the here presented study [P6] instead applies a more realistic bit error model to the encoded traffic, including the coding overhead. We present formal equations to calculate the impact in terms of frame error rate and packet loss rate before and after decoding. A trace-based simulation highlights the implications for relevant real-world applications, namely QUIC web browsing, UFTP file transfer, Skype VoIP, and FFmpeg video streaming. Figure 31 depicts one of the core results, namely the packet reception rate after decoding. The shown heatmap verifies our hypothesis, that the coding overhead has a significant impact on the overall packet loss rate if a bit error model is applied, In some cases, this negative impact is so large that it even overshadows the coding gains, even if additional redundancy is invested.

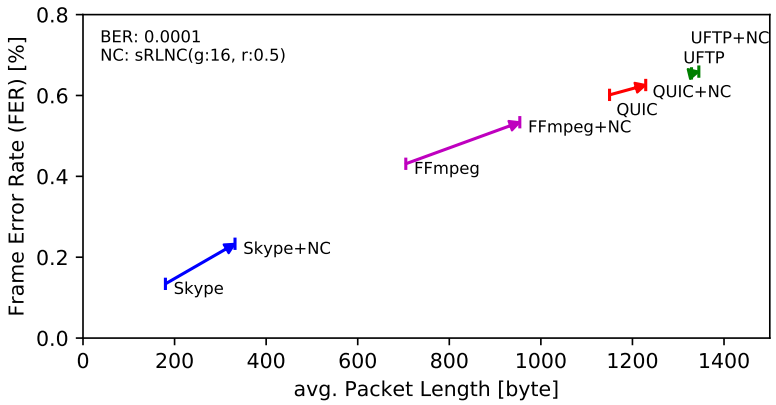


Figure 30: Motivational example: Impact of network coding overhead (sRLNC, g=16, r=0.5) on average frame error rate using a Bernoulli bit error model with avg. BER=0.0001 (c.f. [P6], Figure 1).

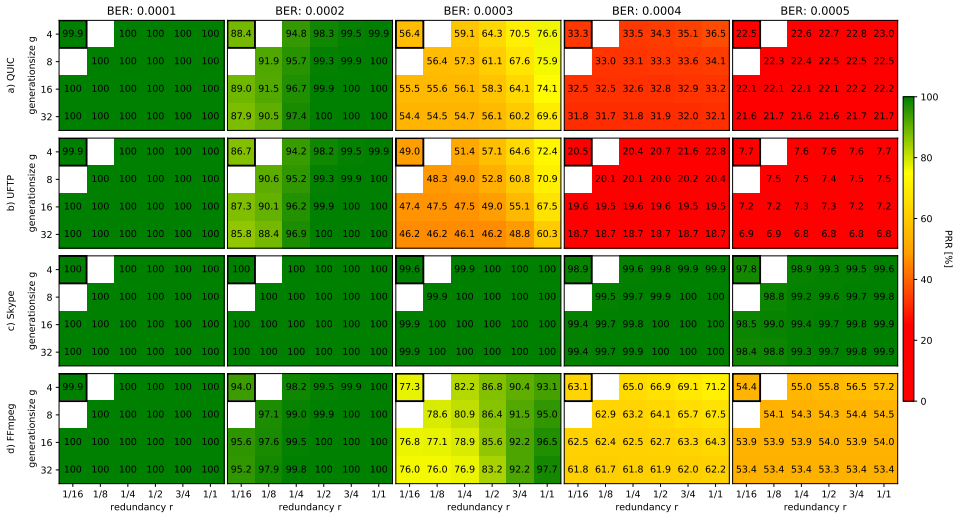


Figure 31: Heatmap of average Packet Reception Rate **after** decoding (PRR_{DEC}) for different sRLNC parametrizations. The uncoded reference PRR is given in each marked top left corner, (c.f. [P6], Figure 1).

Context & Discussion: While [P6] was published after [P4], it is not a direct follow up. Instead, the later study delves deeper into the underlying motivation and highlights the need for padding overhead reduction schemes.In the overall context of this thesis, the most discussable claim in [P6] is the hypothesis that it is more realistic to use a bit-error model instead of a packet length independent loss model, even though all our other publications use the latter one. While bit-error models do indeed better reflect the physical impact of inferences and signal degradation, length independent models are still widely used in networking research. For example, all models in the popular network emulator *netem* simply operate on a per-packet basis. Even *netem's* corruption option does not factor in the packet lengths. This limitation was also inherited by our initial *link'em* bridge prototype, due to its foundation on *netem*. To solve this issue, we have not only created Bernoulli bit error model, but also ported a Signal-to-Noise-Ratio (SNR)-based model from the network simulator $NS-3$ into *link'em* using our custom loss module extension. This enables a proper emulation of bit errors without relying on expensive hardware or non-reliable methods, like the microwave setup by Kim et al. [27]. Using such an approach would be an interesting topic for future work, not only to validate our simulation results of [P4] and [P6], but also evaluate the padding reduction schemes by Compta et al. [8] in a real-world testbed.

4 Conclusion & Future Work

Conclusion: This thesis has summarized and discussed our contributions to field of network coding-based FEC. While each publication delved deeper into its respective topic and solved a distinctive problem, the superordinate goal was to substantially improve the overall state-of-the-art and bring network coding one step closer to practice. Sticking to our motivational *bridging the gap* analogy, which was initially depicted in Figure 1, we have introduced novel methodologies to connect the information theory fundamentals of network coding to its real-world applicability. This was achieved by showing efficient ways to integrate coding, creating better evaluation methods, and shining a light on the impact of padding overhead for real-world applications. More precisely, [P1] and [P2] presented efficient ways to make network coding usable within IoT protocols, namely CoAP and MQTT-SN, to overcome packet loss in wireless scenarios. Bot testbed evaluations verified that the coding schemes can significantly reduce the completion time of a reliable file transfer and, thus, save valuable energy. To enable a better evaluation of possible coding gains, [P3] and [P5] introduced novel tools and methods to do so. Thereby, the developed *link'em* bridge [P3] was also featured in the VoIP testbed of [P5] to enable a reliable packet loss emulation. This usage does not only link our publications, but also verifies the practical purpose of the *link'em* bridge for networking research. Finally, in [P4] and [P6], we have highlighted the occurrence of padding overhead for real-world applications with heterogeneous packet length distributions and presented a novel packet-structure preserving coding scheme to overcome the problem.

Future Work: The presented contributions of this thesis did not only solve specific relevant challenges, but also paved the way for future research. While our publications have introduced efficient network coding extension for CoAP [P1] and MQTT-SN [P2], yet, it would be interesting to adapt our solutions to other protocols. To give a specific example, especially an extension for the Data Distribution Service (DDS) looks promising. Because DDS implements a data-centric publish-subscribe architecture and stores the data in a distributed manner between all nodes, promising coding opportunities arise. Not only can our network coding enhanced publish-subscribe pattern from [P2] speed up the data transfer, but the data blocks them self can also be distributed as independent linear combinations across the network. Such an approach solves the coupon collector problem, similar to the network coded file-sharing in Avalanche [15]. This does not only make the data

distribution process much easier, but also increases the overall robustness of the system.

The IETF Coding for efficient NetWork Communications Research Group (NWCRG) currently works on a draft to add packet-level FEC to QUIC, similar to TETRYS' sliding window encoding scheme [42]. Such an integration would bring network coding-based FEC to a vast variety of applications, which makes it necessary to address the challenge of padding overhead reduction. Especially our findings in [P6] highlight, how strong the negative impact of this issue really is. While our proposed size-based coding scheme [P4] is one solution to the problem, a real-world comparison to other approaches would be interesting, e.g., the fragmentation schemes by Compta et al. [8].

Considering *SPQER* [P5], we believe that the approach has great potential, especially with to the ongoing push towards better machine learning algorithms in mind. Taking a look at the related field of video streaming QoE, VMAF, a machine learning-based approach has replaced purely image-based metrics like SSIM, and is now considered state-of-art. A similar change will probably happen to VoIP QoE metrics. The results in our journal publication were calculated using a classifier, which was trained on a dataset of carefully created annotations. This allows the classifier to achieve a high score in correspondence to the ground truth, but does not directly reflect the true human hearing capability. Thus, by training a specialized classifier for *SPQER*, which does maybe not achieve a perfect score compared to the annotation data set, but instead better mimics the human reception of speech, a very well fitting VoIP QoE could be calculated. While such developments are still subject to long-term future work, our *link'em* bridge [P3] has already shown its worth. Not only was it used in our *SPQER* testbed, but the bridge was also featured in the setup in [30] to create specific frame drop patterns within an RTP video stream. Such an adaption does not only highlight the versatility of *link'em* and shows its potential for future work, it also verifies that our contributions are not limited to network coding-based FEC, but can also be applied to other areas of networking as well.

Thus, to conclude, this thesis does not only make meaningful contributions to specific network coding challenges and bridge the gap between information theory and real-world usage, but also enables future work, even in further areas of networking research.

References

[1] R. Ahlswede, N. Cai, S.Y. Li, and R.W. Yeung, *"Network information flow"*, IEEE Transactions on Information Theory, 46(4), IEEE, 2000.

[2] J. Ahrenholz, C. Danilov, T.R. Henderson, and J.H. Kim, *"CORE: A real-time network emulator"*, IEEE Military Communications Conference (MILCOM), IEEE, 2008.

[3] B.S. Bloom, M.D. Engelhart, E.J. Furst, W.H. Hill, and D.R. Krathwohl, "Taxonomy of educational objectives: The classification of educational goals. Handbook I: Cognitive domain", David McKay Company, New York, 1956.

[4] J. Baylis, *"Error Correcting Codes: A Mathematical Introduction"*, Routledge, 2018.

[5] J. Bauer, B. Siegmann, T. Jarmer, and N. Aschenbruck, *"On the Potential of Wireless Sensor Networks for the In-Situ Assessment of Crop Leaf Area Index"*, Computers and Electronics in Agriculture, 128, Elsevier, 2016.

[6] J. Bauer, and N. Aschenbruck, *"Measuring and adapting MQTT in cellular networks for collaborative smart farming"*, IEEE Conference on Local Computer Networks (LCN), IEEE, 2017.

[7] C. Berrou, A. Glavieux, A., and P. Thitimajshima, *"Near Shannon limit error-correcting coding and decoding: Turbo-codes"*, IEEE International Conference on Communications (ICC), IEEE, 1993.

[8] P.T. Compta, F.H. Fitzek, and D.E. Lucani, *"On the effects of heterogeneous packet lengths on network coding"*, European Wireless, VDE, 2014.

[9] P.T. Compta, F.H. Fitzek, and D.E. Lucani, *"Network coding is the 5G Key Enabling Technology: effects and strategies to manage heterogeneous packet lengths"*, Transactions on Emerging Telecommunications Technologies, 26(1), Wiley, 2015.

[10] J. Detchart, E. Lochin, J. Lacan, and V. Roca, *"Tetrys, an on-the-fly network coding protocol"*, NWCRG Internet-Draft, expired 06.09.2018.

[11] H.M. Edwards, *"Galois Theory"*, Graduate Texts in Mathematics, Springer, 1993.

[12] R. Gallager, *"Low-density parity-check codes"*, IRE Transactions on Information Theory, IRE, 1962.

[13] R.R. Fontes, S. Afzal, S.H. Brito, M.A. Santos, and C.E. Rothenberg, *"Mininet-WiFi: Emulating software-defined wireless networks"*. International Conference on Network and Service Management (CNSM), IEEE, 2015.

[14] O. Gazi, *"Forward Error Correction via Channel Coding"*, Springer, 2020.

[15] C. Gkantsidis, and P.R. Rodriguez. *"Network coding for large scale content distribution"*, Joint Conference of the IEEE Computer and Communications Societies, 24 (4), IEEE, 2005.

[16] G. Hasslinger, and O. Hohlfeld, *"The Gilbert-Elliott Model for Packet Loss in Real Time Services on the Internet"*, GI/ITG Conference on Measurement, Modelling and Evalutation of Computer and Communication Systems, VDE, 2008.

[17] J. Heide, M.V. Pedersen, F.H. Fitzek, and M. Médard, *"On code parameters and coding vector representation for practical RLNC"*, IEEE international conference on communications (ICC), IEEE, 2011.

[18] A. Hines, J. Skoglund, A. C. Kokaram, and N. Harte, *"ViSQOL: an objective speech quality model"*, EURASIP Journal on Audio, Speech, and Music Processing 13, 2015.

[19] T. Ho, M. Médard, 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, 52(10), IEEE, 2006.

[20] M. Hundebøll, J. Ledet-Pedersen, J. Heide, M.V. Pedersen, S.A. Rein, and F.H. Fitzek, *"Catwoman: Implementation and performance evaluation of ieee 802.11 based multi-hop networks using network coding"*, IEEE Vehicular Technology Conference, IEEE, 2012.

[21] ITU-T, *"P.862: Perceptual evaluation of speech quality (PESQ)"*, Recommendation P.862, International Telecommunication Union, Geneva, 2001.

[22] ITU-T, *"P.862.1: Mapping function for transforming p.862 raw result scores to MOS-LQO"*, Recommendation P.862.1, International Telecommunication Union, Geneva, 2003.

[23] ITU-T, *"P.863: Perceptual objective listening quality prediction"*, Recommendation P.863, International Telecommunication Union, Geneva, 2018.

[24] A. Jurgelionis, L.P. Laulajainen, M. Hirvonen, and A.I. Wang, *"An empirical study of netem network emulation functionalities"*, International Conference on Computer Communications and Networks (ICCCN), IEEE, 2011.

[25] J. M. Kates, and K. H. Arehart, *"The hearing-aid speech quality index (HASQI) version 2"*, Journal of the Audio Engineering Society 62 (3), AES, 2014.

[26] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, and J. Crowcroft, *"XORs in the air: practical wireless network coding"*. IEEE/ACM Transactions on Networking (ToN), IEEE, 2008.

[27] M. Kim, J. Cloud, A. ParandehGheibi, L. Urbina, K. Fouli, D. Leith, and M. Médard, *"Network coded tcp (ctcp)"*, arXiv preprint arXiv:1212.2291, 2012.

[28] J. Korhonen, and Y. Wang, *"Effect of packet size on loss rate and delay in wireless links"*, IEEE Wireless Communications and Networking Conference, IEEE, 2005.

[29] J.F. Kurose, K.W. Ross, *"Computer Networking: A Top-Down Approach"*, Addison Wesley, 1999.

[30] D. Laniewski, B.Schütz, N.Aschenbruck, *"On the Impact of Burst Loss for QoE-Based Performance Evaluations for Video Streaming"*, Symposium of the IEEE Conference on Local Computer Networks (LCN), IEEE, 2020.

[31] S.C. Liew, S. Zhang, and L. Lu, *"Physical-layer network coding: Tutorial, survey, and beyond"*. Physical Communication, 6, Elsevier, 2013.

[32] P. Le Callet, S. Möller, and A. Perkis, *"Qualinet white paper on definitions of quality of experience"*, European network on quality of experience in multimedia systems and services (COST Action IC 1003), 2012.

[33] S. Li, R. Yeung, and N. Cai, *"Linear Network Coding"*, IEEE Transactions on Information Theory, 49(2), IEEE, 2003.

[34] L. Lu, and S.C. Liew, *"Asynchronous physical-layer network coding"*, IEEE Transactions on Wireless Communications 11 (2), IEE, 2011.

[35] D.E. Lucani, MStojanovic, and M. Médard, *"Random linear network coding for time division duplexing: When to stop talking and start listening"*, IEEE INFOCOM, IEEE, 2009.

[36] D.E. Lucani, M.V. Pedersen, D. Ruano, C.W. Sørensen, F.H. Fitzek, J. Heide, and O. Geil, *"Fulcrum network codes: A code for fluid allocation of complexity"*, arXiv:1404.6620, 2014.

[37] W. Marslen-Wilson, and L.K. Tyler, *"The temporal structure of spoken language understanding"*, Cognition 8 (1), Elsevier, 1980.

[38] F. Michel, Q. De Coninck, and O. Bonaventure, *"QUIC-FEC: Bringing the benefits of Forward Erasure Correction to QUIC"*, IFIP Networking Conference, IEEE, 2019.

[39] S. Möller, A. Raake, *"Quality of experience: advanced concepts, applications and methods"*, Springer, 2014.

[40] C.Z. Shelby, K. Hartke, C. Bormann, *"The Constrained Application Protocol (CoAP)"*, RFC 7252, IETF CoRE Working Group, work in progress, June 2014.

[41] D. Silva, W. Zeng, and F.R. Kschischang, *"Sparse network coding with overlapping classes"*, In 2009 Workshop on Network Coding, Theory, and Applications, IEEE, 2009.

[42] I. Swett, M.J. Montpetit, V. Roca, and F. Michel, *"Coding for QUIC"*, NWCRG Internet-Draft, expires 10.09.2020.

[43] M. Tuexen, and R. Stewart. *"UDP encapsulation of Stream Control Transmission Protocol (SCTP) packets for end-host to end-host communication"*, RFC 6951, work in progress, May 2013.

[44] I.H. Hou, Y.E. Tsai, T.F. Abdelzaher, and I. Gupta, *"Adapcode: Adaptive network coding for code updates in wireless sensor networks"*, IEEE INFOCOM, IEEE, 2008.

[45] B. Schuetz, and N. Aschenbruck, *"Packet-Preserving Network Coding Schemes for Padding Overhead Reduction"*, IEEE Conference on Local Computer Networks (LCN), IEEE, 2019.

[46] A.Stanford-Clark, and H.L. Truong, *"MQTT for Sensor Networks (MQTT-SN) Protocol Specification Version 1.2"*, IBM, November 2013.

[47] I.N. Stewart, *"Galois theory"*, CRC Press, 2015.

[48] J.K. Sundararajan, D. Shah, M. Médard, M. Mitzenmacher, and J. Barros, *"Network coding meets TCP"*, IEEE INFOCOM, IEEE, 2009.

[49] J.K. Sundararajan, D. Shah, M. Médard, S. Jakubczak, M. Mitzenmacher, and J. Barros, *"Network coding meets TCP: Theory and implementation"*, Proceedings of the IEEE, 99(3), 2011.

[50] , M. Taghouti, D.E. Lucani, M.V. Pedersen, and A. Bouallegue, *On the impact of zero-padding in network coding efficiency with internet traffic and video traces*, VDE European Wireless Conference, VDE, 2016.

[51] , M. Taghouti, D.E. Lucani, M.V. Pedersen, and A. Bouallegue, *Random linear network coding for streams with unequally sized packets: Overhead reduction without zero-padded schemes*, IEEE International Conference on-Telecommunications (ICT), IEEE, 2016.

[52] M. Taghouti, D.E. Lucani, J.A. Cabrera, M. Reisslein, M.V. Pedersen, F.H. Fitzek, *"Reduction of padding overhead for RLNC media distribution with variable size packets"*, IEEE Transactions on Broadcasting, IEEE, 2019.

[53] T.T. Thai, J. Lacan, and E. Lochin, *"Joint on-the-fly network coding/video quality adaptation for real-time delivery"*, Signal Processing: Image Communication, 29(4), Elsevier, 2014.

[54] I. Wechsung, and K. De Moor, *"Quality of experience versus user experience"*, Quality of experience, Springer, 2014.

[55] S. Zhang, S.C. Liew, and P.P. Lam, *"Hot topic: Physical-layer network coding"*, Proceedings of the 12th annual international conference on mobile computing and networking (MobiCom), ACM, 2006.

Page intentionally left blank.

Attached Publications

Notice: *Due to copyright protection, the publications are not attached to this digital version but can be obtained using the following DOI links to the publishers websites. The printed version of this thesis is self-contained and can be found in the library of the University of Osnabrück.*

[P1] Bertram Schütz, Nils Aschenbruck
"Adding a Network Coding Extension to CoAP for Large Resource Transfer", 41st IEEE Conference on Local Computer Networks (LCN), Dubai, UAE, November 2016.
DOI: 10.1109/LCN.2016.122

[P2] Bertram Schütz, Jan Bauer, Nils Aschenbruck
"Improving Energy Efficiency of MQTT-SN in Lossy Environments Using Seed-Based Network Coding", 42nd IEEE Conference on Local Computer Networks (LCN), Singapore, October 2017.
DOI: 10.1109/LCN.2017.87

[P3] Bertram Schütz, Stefanie Thieme, Nils Aschenbruck,
Leonhard Brüggemann, Alexander Ditt, Dominic Laniewski, Dennis Rieke
"Link 'em: An Open Source Link Emulation Bridge for Reproducible Networking Research", International Conference on Networked Systems (NetSys), Munich, Germany, March 2019.
DOI: 10.1109/NetSys.2019.8854509

[P4] Bertram Schütz, Nils Aschenbruck
"Packet-Preserving Network Coding Schemes for Padding Overhead Reduction", 44th IEEE Conference on Local Computer Networks (LCN), Osnabrück, Germany, October 2019.
DOI: 10.1109/LCN44214.2019.8990879

[P5] Bertram Schütz, Nils Aschenbruck
"SPQER: Speech Quality Evaluation using Word Recognition for VoIP Communication in Lossy and Mobile Networks", IEEE Open Journal of the Computer Society, July 2020.
DOI: 10.1109/OJCS.2020.3011392

[P6] Bertram Schütz, Nils Aschenbruck
"The Impact of Bit Errors on Intra-Session Network Coding with Heterogeneous Packet Lengths", Symposium of the 45th IEEE Conference on Local Computer Networks (LCN), Sydney, Australia, October 2020.
DOI: 10.1109/LCNSymposium50271.2020.9363259