Rate-adaptive Link Quality Estimation for Coded Packet Networks

Maurice Leclaire*, Stephan M. Günther*, Marten Lienen*, Maximilian Riemensberger†, Georg Carle*
*Chair of Network Architectures and Services, Department of Informatics
†Methoden der Signalverarbeitung, Department of Electrical and Computer Engineering
Technical University of Munich, Germany
Email: {maurice.leclaire, guenther, marten.lienen, carle, riemensberger}@tum.de

***Abstract*—Coded packet networks allow for proactive injection of redundant packets to compensate for packet loss. Link metrics are usually based on the estimated transmission counter (ETX). This metric is used to determine the expected number of coded packets needed, but does not make guarantees for a specific decoding probability. In this paper we show that relying on the ETX metric leads to a surprisingly high probability that decoding is not possible. Based on this result, we derive a redundancy scheme to allow for an adjustable decoding probability. In a third step, we extend this scheme to also consider the reliability of link quality estimates themselves. We provide a numerically stable and hardware-accelerated implementation of our redundancy scheme [1], and compare all approaches in a simulated environment. Finally, we show the effect of the new redundancy scheme on different transport layer protocols in a wireless setup with random linear network coding.**

***Index Terms*—wireless, link quality, network coding**

I. Introduction and related work

In packet networks, both erasure coding [2, 3] and network coding [4] allow to compensate for packet loss by proactively injecting redundant coded packets. Under these conditions, erasure coding can be considered a special case of network coding, i. e., coding is done on a point-to-point basis, which can be both an end-to-end connection over multiple hops or decoding and recoding over single hops. Network coding additionally allows intermediate nodes to form coded packets without the need for decoding, or even to code on packets belonging to different flows. Coding is therefore not only performed at a source node but by all nodes, which is why one can say *the network is coding*.

Both erasure and network coding require to determine the correct amount of redundancy to allow the destination to decode while avoiding an excessive amount of superfluous transmissions. This turns out to be a difficult task in practice. Considering a simple two-node wireless network, any kind of network coding can be considered as a packet-level erasure code. Assuming random linear coding, i. e., coded packets are random linear combinations of source packets, estimating the necessary number of coded packets consists of two steps: estimate the (asymmetric) link quality and derive the necessary amount of redundancy. Each packet that is overheard results in a noisy reading of the link quality. This can be the number of lost packets since the last one received or the signal-to-noise ratio (SNR) during reception. A common method to estimate the actual link quality while accounting for the fading information content of past transmissions is an exponentially weighted moving average (EWMA) based on the total amount of overheard and missed packets. Woo and Culler [5] introduce the more elaborate EWMA with window mean (WMEWMA) which computes the average success rate over fixed time intervals and updates the actual estimator at the end of each interval. This is essentially a low-pass filter for the ordinary EWMA and makes the estimator independent from the packet rate. Based on an estimate of the packet success rate, the ETX metric [6] computes the expected number of transmissions until success. If the sender stops transmitting immediately after the transmission was successful, e. g. if it is notified via acknowledgements, the ETX metric yields the average number of transmissions necessary to transfer a single packet. This approach is widely used in wireless routing protocols such as Babel [7], B.A.T.M.A.N [8], and (optionally) OLSRv2 [9]. The ETX metric is also commonly used to compute the required redundancy in coded packet networks, for instance by MORE [10, 11], COPE [12], and CORE [13]. However, transmitting the corresponding number of coded packets does not guarantee any specific probability for successful decoding at the destination.

The two parts — estimating the link quality and predicting redundancy — are the basis for efficient coded packet networks. However, the commonly used schemes to predict the redundancy only rely on the actual estimate for the link quality but do not take the reliability of that estimate itself into account, i. e., only a single value (the actual estimate) is used to predict the amount of redundant packets. Renner et al. [14] show in the context of wireless sensor networks that this is insufficient and propose the Holistic Packet Statistics (HoPS) metric, which consists of four estimates (short-term and long-term averages, deviation of the short-term vs. long-term average, and a trend indicator), but it is left open how those values should be used. Koksal and Balakrishnan [15] introduce another estimator that takes the standard deviation of the link quality estimates into account. Although both of these approaches incorporate the stability of the link, they do not consider the quality of the estimates.

In this paper we propose a time-variant link quality estimator and a quality aware redundancy scheme that allow to consider the reliability of the actual link estimates. We start in Section II with the definition of our system model and derive the time-

variant estimator for the link quality. Based on this we introduce three different redundancy schemes. Scheme 1 determines the expected number of transmissions for a generation of N packets to allow for decoding at the destination, which is equivalent to the ETX metric over a single hop. We show that this approach exhibits a surprisingly high probability that the destination is unable to decode. Based on this result we derive Scheme 2 that allows for an adjustable decoding probability, which can be considered a confidence level. Therefore, we refer to Scheme 2 as *confidence level-based redundancy scheme*. Both schemes are derived and discussed in Section III.

Our main contribution consists in Scheme 3 that is derived in Section IV. In contrast to Scheme 1 and 2 it incorporates the reliability of the actual link estimate itself, which is why we refer to Scheme 3 as *quality-aware redundancy scheme*. We further introduce a numerically stable and hardware-accelerated implementation of Scheme 3 in Section V, which is provided for download at [1]. In Section VI, we first compare the different approaches in a simulated environment and afterwards show the influences of Scheme 3 on transport layer protocols in a real-world wireless testbed using random linear network coding in Section VI. Section VII concludes the paper.

II. System model

We consider a coded packet erasure network $G = (V, A)$ consisting of two nodes $V = \{v, w\}$ and two arcs $A = \{(v, w), (w, v)\}$ representing an asymmetric link of time-variant reliability. Packet erasures are assumed to be independently and identically distributed over short periods of time but may vary in the long term. Without loss of generality we assume that node v is the transmitter and node w the receiver. Packets[1] transmitted are considered to carry a unique sequence number starting at zero. If packets are transmitted in order, i. e., no re-ordering occurs after generating the sequence number, the receiver can thus determine the number of missed packets between two successfully overheard packets.

Let Z_k denote a geometrically distributed random variable counting the number of unsuccessful transmissions by v after the $(k-1)$-th and before k-th packet are overheard by w. The vector $\boldsymbol{Z}_k = [Z_1, \ldots, Z_k]^T$ represents all sequences of events leading to k successfully overheard packets at w. After receiving the k-th packet, w has a concrete observation $\boldsymbol{z}_k = [z_1, \ldots, z_k]^T$ of $\boldsymbol{Z}_k$. From that it can derive the number of successfully overheard packets $p_k = \dim \boldsymbol{z}_k = k$ and missed packets $q_k = \mathbf{1}^T \boldsymbol{z}_k$.

Assuming random linear network coding, the transmitter may code on a subset of packets, which we refer to as a generation of N packets. Packet erasures may be compensated by an arbitrary linear combination of those packets that is linearly independent of the set of packets already received. If coding is done over a sufficiently large Galois field, we can neglect random linear dependencies between the first N packets received per generation. For instance, in case of $\mathrm{GF}(256)$ there is a $99.61\,\%$ probability that N randomly generated coded packets are linearly independent [16, 17]. Otherwise, the following discussions can easily be extended to account for random linear dependencies if necessary. If the link quality is known in advance, transmitters can inject redundant packets proactively to compensate for losses. The link quality estimate, i. e. the success rate ρ of the first k packet transmissions, as given by

$$\rho = \lim_{k \to \infty} \frac{p_k}{p_k + q_k} \tag{1}$$

converges to the expectation of the success probability. Equation 1 does not yet account for time-variant erasure probabilities. For this purpose time-variant estimators such as the exponentially weighted moving average (EWMA) or window mean EWMA [5] estimators are commonly used.

We propose a modified version: Let t_k denote the point in time when the k-th packet is being received. Given a time constant $\tau > 0$, we define the time-variant functions p_k and q_k as

$$p_k(t) = (p_{k-1}(t_k) + 1)\, e^{-\tau(t - t_k)}, \tag{2}$$

$$q_k(t) = (q_{k-1}(t_k) + z_k)\, e^{-\tau(t - t_k)}, \tag{3}$$

with initial values $p_0 = q_0 = 0$. These functions combine the actual counts of overheard and missed packets with an exponential decay and thus represent a time-variant packet count. The time constant τ determines how fast both expressions tend to zero when no packets are being received. Note that (2) and (3) are equivalent to an exponentially weighted moving average if packets arrive in constant time intervals[2]. Analogous to (1) we use

$$\rho_k(t) = \frac{p_k(t)}{p_k(t) + q_k(t)} \tag{4}$$

as time-variant estimator for the link quality. If not noted otherwise, we abbreviate (2), (3), and (4) by simply writing $p = \lfloor p_k(t) \rfloor$, $q = \lfloor q_k(t) \rfloor$, and $\rho = \rho_k(t)$, respectively. Note that the floor function ensures that p and q are integer values, which is necessary for the following discussion.

The estimator as described above allows the receiver w to estimate the success probability on link $(v, w) \in A$. However, the estimate is needed by v to determine the amount of redundancy that should be injected. Therefore, we assume that w includes the actual values of p and q in beacon traffic which is broadcast periodically. This approach naturally extends to multiple nodes, i. e. all $w \in V$ that are in range of transmitter v can estimate the link quality ρ_{vw}, which is communicated back to v through regular beacon traffic. Note that we do not consider multi-hop networks for the scope of this paper, i. e. we are solely concerned with the link between two neighboring nodes.

Based on these assumptions we now derive different schemes to determine the minimum number $n \geq N$ of linearly independent packets to allow for decoding. To this end we introduce the random variable X counting the number of

[1] As we are considering transmission from the view point of the link layer, we are actually talking about frames, which is however an unusual term in the scope of network coding.

[2] For the convex weight we obtain $\alpha = e^{\tau}$.

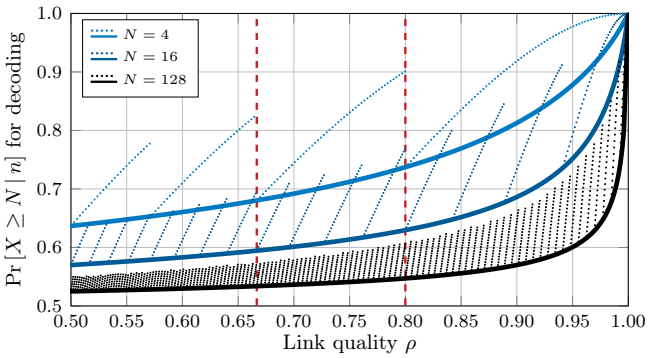

Figure 1. Decoding probability $\Pr\left[X \geq N \mid n_1^*\right]$ depending on the link quality ρ for generation sizes $N \in \{4, 16, 128\}$. Solid plots assume that n is real number while dotted plots indicate decoding probabilities if n is restricted to integral values. The vertical dashed (red) lines indicate the range of values for ρ such that $n_1^* = \lceil 4/\rho \rceil = 6$, i. e., six coded packets are sent for a generation of size $N = 4$.

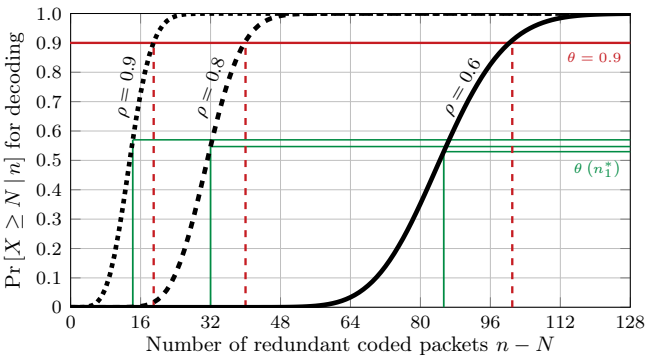

Figure 2. Decoding probability $\Pr[X \geq N \mid n]$ if $n - N$ redundant packets are transmitted for a generation size of $N = 128$ and link qualities of $\rho \in \{0.6, 0.8, 0.9\}$. Decoding probabilities $\theta\left(n_1^*\right)$ for Scheme 1 are marked by the solid thin (green) lines. The confidence level $\theta = 0.9$ is indicated by the solid thick (red) line and the corresponding number of redundant packets $n_2^* - N$ according to Scheme 2 are highlighted by vertical dashed (red) lines.

coded packets received for the current generation given that $n \geq 0$ coded packets have been transmitted. Then X follows a binomial distribution with parameters n and ρ.

III. Quality-unaware redundancy schemes

We refer to schemes determining the necessary amount of redundant coded packets without taking the quality of link estimates into account as *quality-unaware* schemes. First, we consider the de-facto standard scheme based on the expected number of redundant packets and discuss its weaknesses. Afterwards, we extend that scheme to allow for additional control over redundancy using confidence levels.

A. Expectation-based (Scheme 1)

Given X counting the number of coded packets received for a given generation as introduced in Section II, we have that $\mathrm{E}[X] = n\rho_k$. Since we need N coded packets for decoding, we require that at least that many are received in expectation:

$$\mathrm{E}[X] = n\rho_k \geq N \quad \Leftrightarrow \quad n \geq \frac{N}{\rho_k}.$$

Given that no fractions of packets can be sent, $n_1^* = \lceil N/\rho_k \rceil$ is a lower bound for the number of coded packets that have to be transmitted. The widely [7–13] used ETX metric is based on this approach.

However, the probability that n_1^* packets are sufficient for decoding is surprisingly low as shown in Figure 1. The thick solid plots indicate the exact values for n if fractions of packets could be sent, i. e., if n was a real number. The data points (dotted plots) indicate the actual decoding probabilities as n_1^* is limited to integral values only. For instance, consider a generation size of $N = 4$. For link qualities $0.\overline{6} \leq \rho < 0.8$ marked by the two vertical dashed (red) lines in Figure 1, $n_1^* = 6$ coded packets would be transmitted. However, if the link quality becomes equal to or greater than 0.8, one less coded packet is transmitted which in turn causes an abrupt drop of the decoding probability.

These results suggest that transmitting n_1^* coded packets is insufficient for decoding from a practical point of view: even for high quality links with $\rho > 0.9$, decoding becomes only possible after sending feedback and waiting for additional coded packets for a significant fraction of generations. In case of random linear coding, this induces delays of at least one round trip time between source and destination. Moreover, small changes of the link quality can have an unproportional large influence on decoding probability. In addition, the decoding probability tends to further reduce with larger generations.

B. Confidence level-based (Scheme 2)

Given a confidence level $0 < \theta < 1$, we are interested in determining

$$n_2^* = \min n \geq N : \Pr[X \geq N \mid n] \geq \theta, \tag{5}$$

i. e., determining the minimum number n_2^* of coded packets necessary such that the decoding probability is at least θ. The confidence level θ thus gives additional control over the amount of redundancy being injected by allowing to inject more packets than needed on average to reduce decoding delays due to waiting for feedback from the receiver.

The dotted, dashed, and solid (black) plots in Figure 2 show the decoding probability depending on the number of redundant coded packets for a generation size of $N = 128$ at link qualities of $\rho \in \{0.6, 0.8, 0.9\}$. Sending no redundant packets at all is obviously not enough for imperfect links, and decoding probability is close to zero. The thin solid (green) lines indicate the number of redundant packets $n_1^* - N$ that would be injected using Scheme 1 as described in Section III-A and the corresponding decoding probabilities $\theta\left(n_1^*\right)$. For instance, at $\rho = 0.9$ (dotted plot) a total of 15 redundant packets are sent, which results in a decoding probability of $\theta \approx 57\,\%$. The solid horizontal (red) line in Figure 2 marks a decoding probability of $\theta = 0.9$. The vertical dashed (red) lines highlight the corresponding number of redundant packets $n_2^* - N$ needed to achieve $\theta = 0.9$ for the three different link qualities. For instance, at $\rho = 0.9$ (dotted plot) a total of 19 redundant

Table I. Comparing total number of coded packets as derived from Schemes 1 and 2 ($\theta = 0.9$). The additional overhead compared to Scheme 1 is calculated as $\Delta = n_2^*/n_1^* - 1$.

ρ	N	n_1^*	$\theta = 0.9$		$\theta = 0.99$	
			n_2^*	Δ	n_2^*	Δ
0.6	4	7	9	28.6 %	13	85.7 %
	16	27	32	18.5 %	38	40.7 %
	64	107	119	11.2 %	128	19.6 %
	128	214	230	7.5 %	243	13.6 %
0.8	4	5	6	20.0 %	9	80.0 %
	16	20	23	15.0 %	26	30.0 %
	64	80	87	8.8 %	92	15.0 %
	128	160	168	5.0 %	176	10.0 %
0.9	4	5	5	0.0 %	7	40.0 %
	16	18	20	11.1 %	22	22.2 %
	64	72	76	20.0 %	79	9.7 %
	128	143	147	2.8 %	152	6.3 %

packets are necessary, which are only four packets more than scheme 1 would have sent. Thus, scheme 2 transmits about $2.8\,\%$ more coded packets than scheme 1, but increases the decoding probability from 0.55 to 0.9. For larger generations this overhead is tolerable and decreases further with both larger generations and better link quality (see Table I). For small generations the overhead is proportionally larger, e. g. for $N = 16$ and $\rho = 0.9$ Scheme 1 transmits 18 and Scheme 2 20 coded packets, which translates into an overhead of $11\,\%$.

IV. Quality-aware redundancy scheme

As the link quality is unknown and time variant, we use ρ_k as introduced in Section II as an estimate. However, neither the expectation-based nor the confidence level-based schemes are able to take the reliability of ρ_k into account. Consider a period of low traffic, e. g. beacon traffic only. Since few samples are available, ρ_k may be a poor estimate of the actual link quality. When a node suddenly starts transmitting, it has to rely on the actual ρ_k to derive the amount of redundant packets. As more and more samples become available, ρ_k eventually approaches the actual link quality. In case that the initial estimate has been worse than the actual link quality, we transmit more coded packets than necessary. This is unfortunate but tolerable as ρ_k approaches the actual link quality. However, in case the initial estimate has been better, we inject fewer coded packets than necessary according to either scheme. As a result, the receiver is unable to decode and has to wait for timed retransmissions. Transport layer protocols such as TCP are sensitive to such delays as they are unaware of the attempts of lower layers to compensate for losses. Instead, TCP may interpret the decoding delay as packet loss and prematurely initiate congestion avoidance. This leads to the following considerations regarding the redundancy factor:

1) Overestimation during periods of low traffic is not harmful if the estimate adapts when links get loaded.
2) Underestimation in contrast leads to erratic decoding delays and may interfere with transport layer protocols once links get loaded.

This motivates to incorporate the reliability of the time-variant estimator ρ_k into a third scheme, which we refer to as quality-aware redundancy scheme.

A. Quality-aware (Scheme 3)

Let $S \in [0, 1]$ denote a continuous random variable resembling the success probability of individual packet transmissions with unknown probability mass function (PMF) $f_S(s)$. Given a concrete observation $\boldsymbol{z}_k$ of $\boldsymbol{Z}_k$ as introduced in Section II, i. e., given the history of how many packets were lost between two consecutive packets overheard, we can use Bayes' theorem and express the conditional for S by

$$\begin{aligned} f_{S|\boldsymbol{z}_k}(s) &= \frac{\Pr\left[\boldsymbol{Z}_k = \boldsymbol{z}_k | S = s\right] f_S(s)}{\Pr\left[\boldsymbol{Z}_k = \boldsymbol{z}_k\right]} \\ &= \frac{\Pr\left[\boldsymbol{Z}_k = \boldsymbol{z}_k | S = s\right] f_S(s)}{\int_0^1 \Pr\left[\boldsymbol{Z}_k = \boldsymbol{z}_k | S = \xi\right] f_S(\xi)\,\mathrm{d}\xi}. \end{aligned} \tag{6}$$

Given an observation $\boldsymbol{z}_k$, we can determine the number of overheard and missed packets p_k and q_k, respectively, as introduced in Section II. To point out their dependency on $\boldsymbol{z}_k$, we write $p(\boldsymbol{z}_k) = p_k$ and $q(\boldsymbol{z}_k) = q_k$. Now we can determine the probability

$$\Pr\left[\boldsymbol{Z}_k = \boldsymbol{z}_k \,|\, S = s\right] = s^{p(\boldsymbol{z}_k)}(1-s)^{q(\boldsymbol{z}_k)} \tag{7}$$

for any $s \in [0, 1]$ *after* making the first observation, i. e., after receiving the first packet. Before that and in absence of any side information there is no reason to assume anything else than S being uniformly distributed on the closed interval $[0, 1]$, i. e., $f_{S|\boldsymbol{z}_0}(s) = 1$ for all $s \in [0, 1]$. After observing $\boldsymbol{z}_1 = [z_1]^T$, we can thus determine

$$\begin{aligned} f_{S|\boldsymbol{z}_1}(s) &= \frac{\Pr\left[Z_1 = z_1 | S = s\right] f_{S|\boldsymbol{z}_0}(s)}{\Pr\left[Z_1 = z_1\right]} \\ &= \frac{s^{p(\boldsymbol{z}_1)}(1-s)^{q(\boldsymbol{z}_1)}}{\int_0^1 \xi^{p(\boldsymbol{z}_1)}(1-\xi)^{q(\boldsymbol{z}_1)}\,\mathrm{d}\xi} \\ &= \frac{s^{p(\boldsymbol{z}_1)}(1-s)^{q(\boldsymbol{z}_1)}}{\mathrm{B}(p(\boldsymbol{z}_1)+1, q(\boldsymbol{z}_1)+1)} \\ &= \mathrm{Beta}(s, p(\boldsymbol{z}_1)+1, q(\boldsymbol{z}_1)+1), \end{aligned} \tag{8}$$

where $\mathrm{Beta}(s, p+1, q+1)$ denotes the PMF of the Beta distribution and $B(p+1, q+1)$ the Beta function. After observing $\boldsymbol{z}_k = [z_1, ..., z_{k-1}, z_k]^T$ we can thus use $f_{S|\boldsymbol{z}_{k-1}}$ as approximation for the unknown distribution f_S and obtain

$$\begin{aligned} f_{S|\boldsymbol{z}_k}(s) &= \frac{\Pr\left[Z_k = z_k | S = s\right] f_{S|\boldsymbol{z}_{k-1}}(s)}{\Pr\left[Z_k = z_k\right]} \\ &= \frac{s^{p(z_k)}(1-s)^{q(z_k)} f_{S|\boldsymbol{z}_{k-1}}(s)}{\int_0^1 \xi^{p(z_k)}(1-\xi)^{q(z_k)} f_{S|\boldsymbol{z}_{k-1}}(\xi)\,\mathrm{d}\xi} \\ &= \frac{s^{p(z_k)}(1-s)^{q(z_k)} \frac{s^{p(\boldsymbol{z}_{k-1})}(1-s)^{q(\boldsymbol{z}_{k-1})}}{\mathrm{B}(p(\boldsymbol{z}_{k-1})+1, q(\boldsymbol{z}_{k-1})+1)}}{\int_0^1 \xi^{p(z_k)}(1-\xi)^{q(z_k)} \frac{\xi^{p(\boldsymbol{z}_{k-1})}(1-\xi)^{q(\boldsymbol{z}_{k-1})}}{\mathrm{B}(p(\boldsymbol{z}_{k-1})+1, q(\boldsymbol{z}_{k-1})+1)}\,\mathrm{d}\xi} \\ &= \frac{s^{p(z_k)+p(\boldsymbol{z}_{k-1})}(1-s)^{q(z_k)+q(\boldsymbol{z}_{k-1})}}{\int_0^1 \xi^{p(z_1)+p(\boldsymbol{z}_{k-1})}(1-\xi)^{q(z_k)+q(\boldsymbol{z}_{k-1})}\,\mathrm{d}\xi} \end{aligned}$$

$$= \frac{s^{p(\boldsymbol{z}_k)}(1-s)^{q(\boldsymbol{z}_k)}}{\int_0^1 \xi^{p(\boldsymbol{z}_k)}(1-\xi)^{q(\boldsymbol{z}_k)}\,\mathrm{d}\xi}$$
$$= \mathrm{Beta}(s, p(\boldsymbol{z}_k)+1, q(\boldsymbol{z}_k)+1). \tag{9}$$

By using this estimate for the success probability's PMF, we consider an optimization problem similar to (5):

$$n_3^* = \min n \geq N:\ \Pr\left[X \geq N \,|\, n, \boldsymbol{z}_k\right] \geq \theta, \text{ with} \tag{10}$$

$$\Pr\left[X \geq N \,|\, n, \boldsymbol{z}_k\right] = 1 - \Pr\left[X < N \,|\, n, \boldsymbol{z}_k\right]$$
$$= 1 - \sum_{i=0}^{N-1} \Pr\left[X = i \,|\, n, \boldsymbol{z}_k\right]. \tag{11}$$

The new optimal value as obtained by Scheme 3 is denoted by n_3^*. Given a specific realization $\boldsymbol{z}_k$ of $\boldsymbol{Z}_k$ and writing again $p = p(\boldsymbol{z}_k)$ and $q = q(\boldsymbol{z}_k)$, we obtain

$$\Pr\left[X = i \,|\, n, \boldsymbol{z}_k\right] = \int_0^1 \binom{n}{i} s^i (1-s)^{n-i} f_{S|\boldsymbol{z}_k}(s)\,\mathrm{d}s$$
$$= \binom{n}{i} \int_0^1 s^i (1-s)^{n-i} \frac{s^p (1-s)^q}{\mathrm{B}(p+1, q+1)}\,\mathrm{d}s$$
$$= \binom{n}{i} \frac{\mathrm{B}(i+p+1, n-i+q+1)}{\mathrm{B}(p+1, q+1)}. \tag{12}$$

Note that p, q are integral values as defined in (2) and (3). This allows us to express the Beta function $\mathrm{B}(x, y)$ as fraction of factorials, i. e.,

$$\mathrm{B}(x, y) = \frac{(x-1)!(y-1)!}{(x+y-1)!}. \tag{13}$$

Substituting (13) in (12) and reordering the products yields

$$\Pr\left[X = i \,|\, n, \boldsymbol{z}_k\right] = \frac{n!(p+q+1)!}{(n+p+q+1)!} \frac{(i+p)!}{i!p!} \frac{(n-i+q)!}{(n-i)!q!}$$
$$= \prod_{j=1}^{n} \frac{j}{p+q+j+1} \prod_{j=1}^{i} \frac{j+p}{j} \prod_{j=1}^{n-i} \frac{j+q}{j}$$
$$= \underbrace{\prod_{j=1}^{i} \frac{p+j}{p+q+j+1}}_{g} \underbrace{\prod_{j=i+1}^{n} \frac{j(q+j-i)}{(p+q+j+1)(j-i)}}_{h}. \tag{14}$$

Note that factorials were not cancelled against each other at will but in a very specific way to avoid overruns and numeric instability:

1) Since the individual factors of g approach 1 for $j \to \infty$, g remains bounded by $(p/(p+q+1))^i < g \leq 1$ at all times.
2) Since the product $g \cdot h$ represents a probability, we find that h is upper bounded by $h < 1/g = ((p+q+1)/p)^i$.

Since multiplication of two floating point values is not prone to cancellations[3] , the only ways for (14) to become numerically unstable are if either $g \to 0$ or $h \to \infty$. Both cases are result of extremely poor links, i. e., $q \gg p$, which is of no practical interest. Applying this result to (11) finally gives the desired probability

$$\Pr\left[X \geq N \,|\, n, \boldsymbol{z}_k\right] = \tag{15}$$
$$1 - \sum_{i=0}^{N-1} \prod_{j=1}^{i} \frac{p+j}{p+q+j+1} \prod_{j=i+1}^{n} \frac{j(q+j-i)}{(p+q+j+1)(j-i)},$$

yielding a numerically stable blue-print for implementation.

Regarding computational complexity the outer sum is bounded by the constant generation size N while the inner products are bounded by the expected number of packets $n \geq N$ that have to be sent. This results in a complexity of $\mathcal{O}(Nn)$, which we show in Section V can be solved in less than $50\,\mu$s for most link qualities of practical interest.

[3]Multiplication of two floats may still lead to subnormal numbers, which may be mapped to numerically zero depending on implementation. We address this issue in Section V.

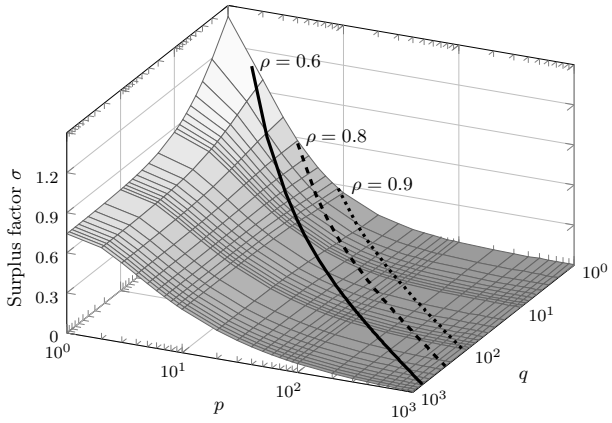

Figure 3. Surplus factor σ depending on p, q for $\theta = 0.9$ and a generation size of $N = 128$, e. g. $\sigma = 0$ means that the confidence level-based and quality aware schemes give the same results. The solid, dashed, and dotted plots indicate the surplus factor for a constant link quality $\rho \in \{0.9, 0.8, 0.6\}$, respectively, and thus show the differences between both schemes at constant link qualities.

B. Differences to the confidence level-based scheme

The unique feature of this new redundancy scheme compared to the ones previously discussed is its ability to take the reliability of the link quality estimate itself into account. If the link is idle, e. g. periodic beacon traffic only, we have only a rough estimate of the actual link quality, which is reflected by small values for the observed packet count $p + q$. Due to that uncertainty more redundancy may be needed to ensure a given confidence level θ. However, as the link gets loaded and packet counters increase, the amount of redundancy should converge to the same amount as determined by Scheme 2.

To assess the differences between n_2^* and n_3^* as determined by Schemes 2 and 3, respectively, we consider surplus factor $\sigma = n_3^*/n_2^* - 1$ of additional redundancy depending on p, q, and θ. The result for $\theta = 0.9$ is shown in Figure 3. We see that the surplus factor σ is large if few samples are available to estimate the link quality. More samples mean that more packets must have been received in the past, i. e., p becomes larger, the estimate for the link quality becomes more reliable, and the surplus factor σ approaches zero. For instance, for $p \geq 1000$ and $q <= p$ we have $\sigma < 0.75\,\%$. In case of $q \gg p$, the

Listing 1. Basic implementation without hardware acceleration. Some casts to double precision floats are omitted for brevity.

```
int update_ralqe(int p, int q, float t, int N) {
    double s[] = {[0 ... N-1] = 1}, sum, g, h;
    int i, n = 1;
    do {
        sum = 0.0;
        g  = (p+n) / (p+q+n+1);
        for (i=0; i<N; i++) {
            h = n*(q+n-i) / ((p+q+n+1)*(n-i));
            if (n<=i)
                s[i] *= g;
            else
                s[i] *= h;
            sum += s[i];
        }
        n++;
    } while ((1.0-sum) < t);
    return n;
}
```

surplus factor σ remains considerable large, which is desired: if p is small, few packets have been received, which does not preclude large values for q in case of bad links. The solid, dashed, and dotted lines in Figure 3 highlight the differences between Schemes 2 and 3 for constant proportions but different absolute values for p, q, i. e., the link quality ρ is constant for each plot.

V. Implementation

Based on the results of Section IV-A, we implement a hardware-accellerated and numerically stable kernel that efficiently solves (15). Both the scalar and vectorized implementations are available for download at [1].

A. Scalar implementation

The scalar implementation is shown in Listing 1. It consists of an array of N elements corresponding to the individual summands of (15). Afterwards, each summand is multiplied by the corresponding factor g or h, which correspond to a single factor of the two products g and h of (14). The result is accumulated in a register within the inner loop. The outer loop increases n by one at a time and checks for the break condition $1 - \Pr[X < N \mid n] \geq \theta$. Note that the inner loop just updates each summand with one new factor at a time, which allows to increment n without solving (15) over and over again. As discussed in Section IV-A, the individual summands remain bounded at all times. However, in case of bad link qualities and small values for p, i. e., $q \gg p$, a summand may become numerically zero when using single precision floats. Therefore, the array containing the individual summands that are updated in each iteration must be double precision.

B. Vectorized implementation

The scalar implementation of Listing 1 can be vectorized and considerably accelerated by using SIMD extensions of modern processors, which we show exemplary by using the AVX[4] [18] instruction set extension introduced with Intel's

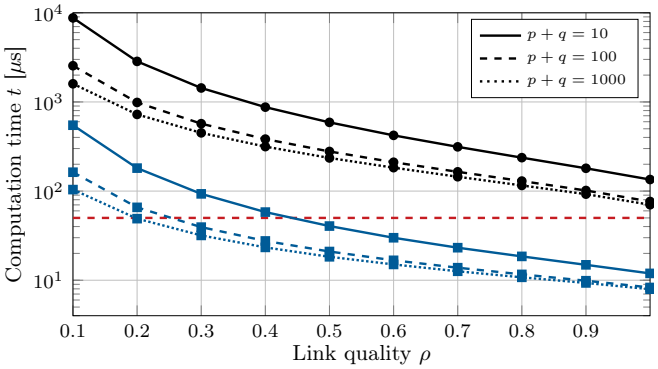

Figure 4. Time to solve (15) depending on the link quality ρ and and different total number of samples $p+q \in \{10, 100, 1000\}$ for a generation of size $N = 256$ and a confidence level of $\theta = 0.99$. Ball markers (black plots) indicate results of the scalar implementation, square marker results for the vectorized implementation (AVX). The dashed line indicates $t = 50\,\mu$s. Both tests were performed on an Intel Core i7-5600U[5]using a single thread only.

Sandy Bridge microarchitecture in 2011. The basic idea is to vectorize the inner loop with a stride size of 8, i. e., single precision floats. The conditional in the inner loop can be replaced by a vector compare of $n \leq \boldsymbol{i}$. The result is a vector indicating the comparison result for each individual component of $\boldsymbol{n}$. This is used in combination with a *blend* operation to construct a vector containg either a product of g or h in each of its components, i. e., h is computed opportunistacally. The combination of opportunistic calculations and blend operations is highly efficient as it removes branches. To avoid the numerical problem mentioned in Section V-A, the resulting vector is converted to double precision before the summands are updated.

C. Computation time

The time needed to solve (15) for a generation of $N = 128$ and a confidence level of $\theta = 0.99$ is shown in Figure 4. The ball markers (black plots) show results for the scalar implementation as printed in Listing 1, square markers (blue plots) indicate results for the vectorized implementation using AVX extensions. We see that the computation time depends on both the link quality ρ as well as the total amount of available samples $p+q$. Both are consequences of the fact that determining $\hat{n}^*$ according to (15) requires to iterate until the confidence level θ is reached. In case of a small number of samples, proportionally more iterations are needed. The dashed (red) line in Figure 4 indicates $t = 50\,\mu$s, i. e., for link qualities better than $\rho = 0.4$ we are able to solve (15) in less than $50\,\mu$s using the vectorized implementation, which accelerates computation by approximately an order of magnitude. Since it is not neccessary to solve $\hat{n}^*$ for each packet being transmitted but instead to updated the value only a couple of times per second or even less, those times are affordable even on smaller devices.

[4] *Advanced Vector Extensions*

[5] Low voltage Broadwell-based notebook processor running at up to 3.2 GHz

VI. Results

First, we validate and compare the different redundancy schemes in a simulated environment. We demonstrate that Scheme 3 converges to Scheme 2 if the link estimate is based on sufficiently large number of samples. Afterwards, we demonstrate the influence of Scheme 3 on transport layer protocols in a wireless network with random linear network coding.

A. Simulation

We simulate the time evolution of all schemes over a timespan of $300\,\mathrm{s}$ using different packet rates and link qualities. The results are shown in Figure 5. The upper part shows the time evolution of $p_k(t)$ and $q_k(t)$. The main part of Figure 5 shows the results of the different redundancy schemes:

1) The dotted (black) plot indicates the redundancy determined by Scheme 1.
2) The dashed (orange) plot represents the result of Scheme 2 for $\theta = 0.9$.
3) The solid (blue) plot indicates redundancy factor n_3^*/N as determined by Scheme 3 for $\theta = 0.9$.
4) The three shaded surface plots in the background indicate the uncertainty as derived from Scheme 3, i. e., $(n_3^* \pm \sqrt{\mathrm{Var}\,[n_3^*]})/N$, at three different confidence levels $\theta \in \{0.5, 0.9, 0.99\}$.
5) The different phases and the parameters (packet rate and link quality) are highlighted by dashed (red) lines.

Since p and q are unbounded above, the link quality estimate as introduced in Section II exhibits an unwanted dependency on the packet rate: a phase of high link utilization results in large values for both packet counters, which in turn take a long time to decrease after the link becomes idle again. In the meantime the link estimate may be considered reliable. A simple solution to that issue is to enforce an upper bound for $p_k + q_k$ whenever packet counters are updated — scaling both counter down when necessary. This approach is legitimate since the surplus factor σ was shown in Section IV-B to be smaller than $0.75\,\%$ for $p \geq 1000$ and $q \leq p$. Consequently, there is not much benefit from leaving p_k and q_k unbounded. For the simulations in this section we limit $p + q \leq 1000$ and choose $\tau = 0.1$ as time constant. This results in a decay of p and q by $63\,\%$ over a time span of $10\,\mathrm{s}$ if no further packet is being received. Decreasing τ leads to a faster decay and thus more up to date estimates, but also increases the variance and thus uncertainty during phases of low traffic.

Phase 1: Simulation starts at $t = 0$ with a packet rate of $5\,\mathrm{packets/s}$ at a link quality of $\theta = 0.9$. It takes approximately $20\,\mathrm{s}$ for all schemes to reach a steady state at which packet counters are approximately $p \approx 44$ and $q \approx 4$. This translates into an estimated link quality of approximately $92\,\%$, which is slightly off the actual quality of $90\,\%$. We clearly see that both Schemes 2 and 3 overestimate the average redundancy as given by Scheme 1. The difference between both schemes stems from the fact that Scheme 3 is aware of the unreliable estimate.

Phase 2: At $t = 60\,\mathrm{s}$ the link quality suddenly drops to $\rho = 0.7$ while the packet rate remains constant at $5\,\mathrm{packets/s}$. All schemes now show some oscillation due to unreliable estimates ranging between $65\,\%$ and $75\,\%$. Decreasing the time constant τ would make the estimate more stable but also require more time to react to changes of the link quality.

Phase 3: At $t = 120\,\mathrm{s}$ the packet rate increases to $100\,\mathrm{packet/s}$ while the link quality remains constant at $\rho = 0.7$ We immediately see a significant increase of p and q, meaning that the estimated link quality is now more reliable. All schemes now show a significantly more stable result. Scheme 3 almost instantly converges to the result of Scheme 2 as it should be the case. In addition, we see that the uncertainty of Scheme 3 as indicated by the shaded surface plots in the background significantly decreases.

Phase 4: At $t = 180\,\mathrm{s}$ the link quality increases to $\rho = 0.9$ while the packet rate remains constant at $100\,\mathrm{packet/s}$. We see that p increases while q decreases proportionally. All schemes adapt accordingly and now show an even more stable behavior.

Phase 5: At $t = 240\,\mathrm{s}$ the packet rate drops to $5\,\mathrm{packet/s}$ while the link quality remains constant at $\rho = 0.9$. We see that all schemes start to oscillate again as the estimate for the link quality becomes more and more unreliable with decreasing p and q. We also see that Scheme 3 starts to diverge from Scheme 2 and falls back to overestimation of redundancy during phases of low traffic.

The behavior of Scheme 3 is desirable in all phases:

1) During phases of low traffic we can afford some additional redundancy as few packets are sent anyway.
2) As soon as the link gets loaded, the amount of redundancy converges to the minimum amount needed to achieve the predetermined confidence level.
3) When the link becomes idle again, the amount of additional redundancy slowly increases over time as the estimate of the link quality becomes more and more unreliable.

Note that convergence does not depend on θ, i. e., for $\theta = 0.99$ and $120 < t \leq 240$ Scheme 2 would be within the upper shaded plot indicating the uncertainty of Scheme 3. It is also interesting to see that for $\theta = 0.5$ Scheme 3 (lower shaded plot) yields results almost identical to Scheme 1.

B. Real-world performance

To demonstrate the potential of quality-aware redundancy in practice, we validate the scheme using our *Network Coding Module (NCM)*[6]. We us a two-node wireless setup as described in Section II. Random linear coding is done on generations of $N = 128$. Only one generation is being used at any time, i. e., the generation window size is set to one and transmission of the next generation cannot start before decoding of the current generation was signaled from the destination back to the source. The link quality was closely monitored during test and was approximately $\rho_{12} = 0.96$ and $\rho_{21} = 0.94$. The

[6]The underlying injection library `libmoep` is available for download at [1]. The NCM will soon be released under GPLv2

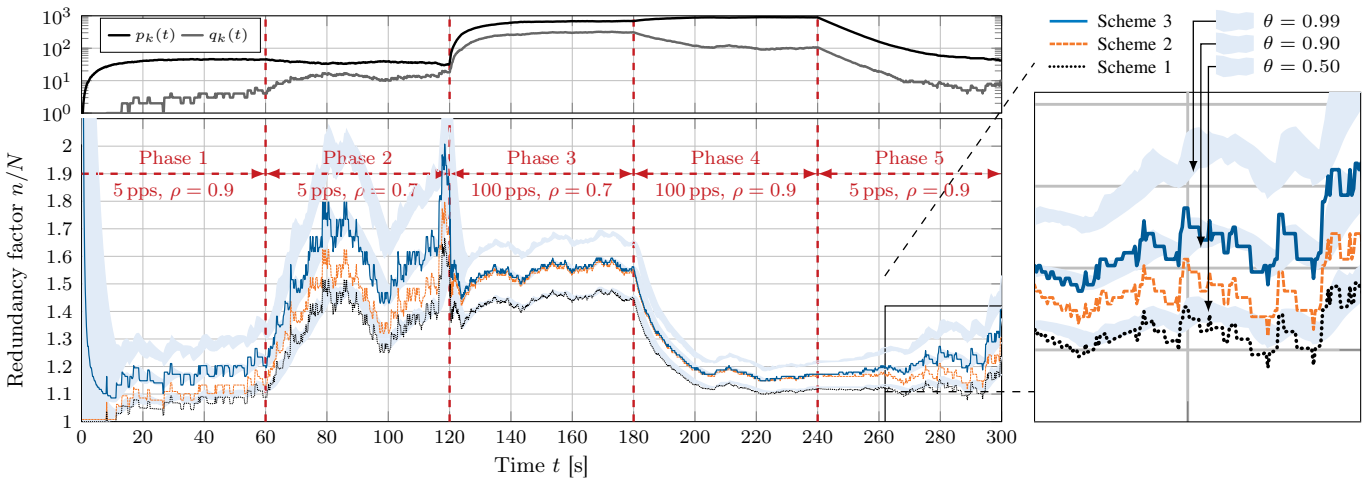

Figure 5. For all simulations we used a generation size of $N = 128$ and a time constant of $\tau = 0.1$ for packet counters. The upper part shows the evolution of p and q for the respective simulation phase. The lower part shows the evolution of the different redundancy schemes: The solid surface plots show a moving average of $n_3^* \pm \sqrt{\mathrm{Var}\left[n_3^*\right]}$ for $\theta \in \{0.5, 0.9, 0.99\}$ derived from the quality-aware redundancy scheme, i. e., wider surface plots indicate more uncertainty with respect to the reliability of n_3^*. The solid, dashed, and dotted plots indicates the time evolution of n_3^*, n_2^*, and n_1^*, respectively. All plots are normalized to the generation size N. The magnification to the right illustrates the different plots.

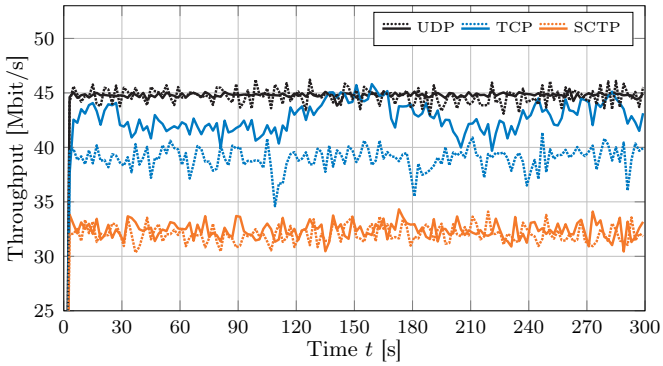

Figure 6. Throughput for a generation of $N = 128$ packets. Solid and dotted plots show the results for Scheme 3 and 1, respectively.

tests were performed using two Qualcomm Atheros AR9580 devices in monitor mode[7]. The MTU was set to 3200 B. The test flows consist of unidirectional data streams sent from node 1 to 2 using TCP, UDP, and SCTP as transport layer protocols. Throughput is measured using `netperfmeter` [19]. The underlying NCM thereby assures that a lossless link is presented to the transport layer protocols.

The results are shown in Figure 6. In case of UDP, which is due to the lack of feedback the only unidirectional protocol from the transport layer's point of view, does not benefit in terms of average throughput, but shows significantly lower oscillations when Scheme 3 is used. In contrast, short-term oscillations are not reduced in case of TCP but average throughput is increased significantly. Interestingly, SCTP shows virtually identical results in terms of throughput (goodput) no matter which redundancy scheme is being used. At this time we did not further investigate this anomaly but a possible explanation is SCTP's built-in semi-reliability, i. e., SCTP schedules retransmissions on its own and gives up after a certain amount of fails.

[7] 5700 MHz, 40 MHz channel width, 800 ns GI, MCS 12, 3200 B MTU

VII. Conclusion and Future Work

Motivated by the prospect that proactive injection of redundancy based on the usual expectation-based scheme causes avoidable decoding delays in practice, we show that this approach allows for decoding with probability of less than 60 %. At this point, decoding is delayed until additional redundant packets arrive. Such "retransmits" usually rely on timeouts or feedback from the destination. The resulting delays are avoidable, and may in practice interfere with transport layer protocols, e. g. TCP's congestion avoidance. This leads to suboptimal link utilization.

Based on this insight, we derive a quality-aware redundancy scheme that takes two important factors into account:

1) determine the necessary redundancy factor to allow for decoding given a certain confidence level, and
2) consider the reliability of the actual link estimate itself.

The new approach is motivated, implemented numerically stable and with support for hardware acceleration, simulated and compared to expectation and confidence level-based schemes, and evaluated in real-world wireless network using random linear network coding. The results show that coded packet networks benefit from our more flexible redundancy scheme. Average TCP throughput is increased by about 20 % while deviation of throughput significantly decreases for UDP flows.

The quality-aware redundancy scheme as introduced in this paper only considers a single link and is therefore limited to network coding scenarios that degenerate to erasure coding. It is left open at this point how the quality-aware redundancy scheme can be extended to multi-hop mesh networks with need for opportunistic- and multipath-aware metrics.

REFERENCES

[1] M. Leclaire, S. Günther, and M. Lienen, "Supplemental material: `ralqe`," http://moep80211.net/plink/lcn2016/.

[2] C. Huitema, *The Case for Packet Level FEC*. Boston, MA: Springer US, 1997, pp. 109–120.

[3] L. Rizzo, "Effective Erasure Codes for Reliable Computer Communication Protocols," ACM SIGCOMM *Computation Communication Review*, vol. 27, no. 2, pp. 24–36, Apr. 1997.

[4] R. Ahlswede, N. Cai, S.-Y. Li, and R. Yeung, "Network Information Flow," *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1204–1216, Jul. 2000.

[5] A. Woo and D. Culler, "Evaluation of Efficient Link Reliability Estimators for Low-power Wireless Networks," Tech. Rep., September 2003.

[6] D. De Couto, D. Aguayo, J. Bicket, and R. Morris, "A High-Throughput Path Metric for Multi-hop Wireless Routing," *Wireless Networks*, vol. 11, no. 4, pp. 419–434, Jul. 2005.

[7] J. Chroboczek, "The Babel Routing Protocol," *RFC6126*, 2011.

[8] D. Johnson, N. Ntlatlapa, and C. Aichele, "Simple Pragmatic Approach to Mesh Routing using BATMAN," *2nd IFIP International Symposium on Wireless Communications and Information Technology in Developing Countries*, Oct. 2008.

[9] T. Clausen, C. Dearlove, P. Jacquet, and U. Herberg, "The Optimized Link State Routing Protocol Version 2," *RFC7181*, 2014.

[10] S. Chachulski, "Trading Structure for Randomness in Wireless Opportunistic Routing," M.Sc. Thesis, Massachusetts Institute of Technology, 2007.

[11] S. Chachulski, M. Jennings, S. Katti, and D. Katabi, "Trading Structure for Randomness in Wireless Opportunistic Routing," *ACM SIGCOMM*, pp. 169–180, 2007.

[12] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, and J. Crowcroft, "XORs in the Air: Practical Wireless Network Coding," *IEEE/ACM Trans. Netw.*, vol. 16, no. 3, pp. 497–510, Jun. 2008.

[13] J. Krigslund, J. Hansen, M. Hundeboll, D. Lucani, and F. Fitzek, "CORE: COPE with MORE in Wireless Meshed Networks," *Vehicular Technology Conference (VTC Spring)*, 2013.

[14] C. Renner, S. Ernst, C. Weyer, and V. Turau, "Prediction accuracy of link-quality estimators," *8th European Conference on Wireless Sensor Networks*, pp. 1–16, Feb. 2011.

[15] C. E. Koksal and H. Balakrishnan, "Quality-Aware Routing Metrics for Time-Varying Wireless Mesh Networks," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 11, pp. 1984–1994, Nov. 2006.

[16] C. Cooper, "On the Distribution of Rank of a Random Matrix over a Finite Field," *Random Struct. Algorithms*, vol. 17, no. 3-4, pp. 197–212, Oct. 2000.

[17] O. Trullols-Cruces, J. Barcelo-Ordinas, and F. M., "Exact Decoding Probability Under Random Linear Network Coding," *IEEE Communications Letters*, vol. 15, no. 1, pp. 67–69, Nov. 2011.

[18] *Intel 64 and IA-32 Architectures Optimization Reference Manual*, Intel, Jul. 2013.

[19] T. Dreibholz, Becke, H. M. Adhari, and E. Rathgeb, "Evaluation of a new Multipath Congestion Control Scheme using the NetPerfMeter Tool-Chain," *IEEE International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, Sep. 2011.