Improving Energy Efficiency of MQTT-SN in Lossy Environments using Seed-based Network Coding

Bertram Schütz, Jan Bauer, Nils Aschenbruck
University of Osnabrück - Institute of Computer Science,
Wachsbleiche 27, 49090 Osnabrück, Germany
{schuetz, bauer, aschenbruck}@uos.de

***Abstract*—This paper presents an energy-efficient solution to overcome packet loss in Wireless Sensor Networks (WSNs) by adding seed-based Random Linear Network Coding to MQTT for Sensor Networks (MQTT-SN). Whereas most sensors integrated in common WSN devices consume little energy, using the radio is costly. To increase battery lifetime, devices try to minimize their radio uptime, while still satisfy timeliness and reliability of delivered data. The proposed approach uses an optimized seed-based intrasession Network Coding scheme for Forward Error Correction to shorten the sensor node's radio uptime, reducing its power consumption. The presented approach is conform to the MQTT-SN specification and, thus, interoperable with existing systems. Since the implementation is based on the application layer, it is seamlessly deployable via Over-The-Air-Programming. 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.**

I. Introduction

The impact of WSNs is continuously growing. Small, cheap sensor nodes are used for intelligent industrial workflows, smart health care monitoring, and Precision Agriculture. Since a reliable power connection is seldom available in real-world deployments and sophisticated energy harvesting technologies are often too costly, battery lifetime is a central constrain for many applications. Recharging or replacing a battery is not always possible or requires expensive manpower. This makes energy-efficiency a central challenge, when deploying and maintaining a WSN.

While most data-gathering sensors, e.g., thermistors, consume little energy, using the network interface and transmitting the sensed data is costly. To save energy, nodes try to stay as long as possible in a low power state with their network interface turned off. To still satisfy timeliness constraints of the collected data, the nodes have to temporary turn on their interface and transmit the queued packets. In this paper, we improve the energy efficiency of temporary sleeping clients in lossy environments by reducing this costly interface uptime. Since the nodes are working under harsh conditions and have only limited transmission power, packet loss happens quite frequently. In order to still guarantee reliable data transfer with the traditional Automatic Repeat reQuest (ARQ) scheme, packets must be costly retransmitted, increasing the interface uptime. This is of particular significance, if a stop-and-wait strategy is used, as it is the case in the Message Queue Telemetry Transport protocol for Sensor Networks (MQTT-SN) [19], which is a WSN specific adaption of the widely used MQTT protocol. To minimize the number of retransmissions, we add loss tolerance based on Forward Error Correction (FEC) to MQTT-SN. Intrasession Network Coding is an approach, that inherits implicit FEC. We will apply a seed-based Network Coding scheme, that is specially suited for WSNs and verify its benefits. Our proposed approach is an application layer solution, which guarantees interoperability and can be seamlessly deployed in existing sensor networks as a software update, e.g., via Over-The-Air-Programming (OTAP).

A. Motivational real-world deployment

The conducted research is motivated by the challenges, we have observed in a real-world crop monitoring WSN deployment in the context of Precision Agriculture. Based on an evaluation of the potential of low-cost sensors for the assessment of relevant crop parameters [3], we designed a specific sensor network for a continuous monitoring of an important and widely-used crop parameter, namely the leaf area index (LAI), which is closely related to the vegetative biomass as it indicates the leaf area per ground area. This sensing system has been deployed in a specific wheat breeding test area, as shown in Figure 1.

For the LAI estimation, two groups of sensor nodes were deployed to measure the light intensity above and below the canopy. The nodes A_n were placed directly in the field, while Node B captures the non-intercepted light intensity

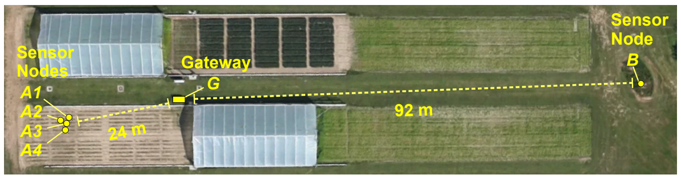

Fig. 1. Motivational WSN deployment. Imagery ©Google, 2017

above the canopy from an elevated position close to the field. Each sensor node transmits its collected data to the central gateway G, which is placed at the border. The gateway then forwards the data to a remote server. There, the measurements are leveraged as decision support in the agricultural process chain to realize a site-specific farm management.

Since the sensor nodes are placed in-field, they should not interfere with vegetation. Thus, they have to be size- and cost-minimized and guarantee a long battery lifetime without maintenance. These fundamental constrains limit the nodes capabilities, especially the radio's transmitting power. Combined with the harsh weather conditions in outdoor environments, the resulting low signal strengths characteristically lead to high packet loss rates. This makes reliable message delivery with minimal energy consumption a common and crucial challenge for low-power and lossy networks, particularly WSNs.

While the presented WSN scenario serves as a motivational use-case, it does not limit the general scope of our approach. The shown solution can be applied to other energy constrained networks as well. For instance, the approach is also promising for proactive maintenance and state monitoring at industrial plants, where sensor nodes at pipes and valves can not always be connected to a power line.

II. Background & Related Work

A. *MQTT for Sensor Networks (MQTT-SN)*

The Message Queue Telemetry Transport protocol for Sensor Networks (MQTT-SN) [19] is a WSN-specific adaptation of the popular IoT protocol MQTT [2], which is widely used in the industry and has recently become an ISO/IEC and OASIS standard [10], [2]. MQTT-SN is designed to enable wireless publish-subscribe communication between constrained sensor nodes. The publish-subscribe pattern is a loose-coupled communication approach that uses so-called topics for individual message exchange. 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 features the core concepts of MQTT, but is designed to better cope with WSN constraints, in particular with limited bandwidth, high packet loss rates, and small maximum transmission units (MTUs).

Similar to MQTT, a common MQTT-SN topology consists of multiple clients and typically a single broker. The main difference is the addition of a specific gateway between the clients and the broker, as shown in Figure 2. If the gateway receives a message from an MQTT-SN client, it performs a mapping to MQTT and forwards the converted message to the broker. The reverse mapping is done for the broker's replies. This mapping is essential, since it does also replace the underlying transport protocol. While the communication between broker and gateway is based on connection-oriented TCP, the clients use MQTT-SN on top of UDP. As UDP is connectionless, the clients do not need to keep a transport layer connection open to communicate with the broker, as it is necessary in regular MQTT. This allows each client to turn off its network interface and enter a sleeping state without

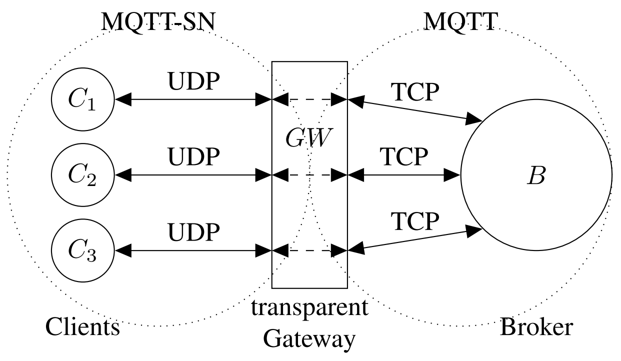

Fig. 2. Transparent MQTT-SN Gateway, cf. [19, Fig. 2]

an extensive reconnection procedure on wake up. The TCP connections to the broker are managed by the gateway for each client. This shifts the energy consumption from the constraint clients to the less energy restricted gateway. Sleeping clients are one of the core concepts, which distinguishes MQTT-SN from MQTT.

The missing transport layer reliability of UDP is compensated on the application layer. Similar to MQTT, predefined Quality of Service (QoS) classes are used. Messages with QoS 0 are sent unreliably in a fire-and-forget manner. QoS 1 corresponds to a stop-and-wait ARQ method. The QoS 2 option guarantees exactly one reception of a message, requiring an additional handshake. Since QoS 2 implies an extensive overhead, it is not further considered. In regular MQTT, the application layer reliability is added on top of TCP's transport layer reliability. For the here presented approach, we see the missing reliability of UDP not as a problem, but as an opportunity. The proposed approach only works, because there is no underlying retransmission scheme. We enhance MQTT-SN by adding a Network Coding FEC scheme. Transport layer reliability, as given by TCP, uses retransmissions and, thus, does not allow FEC on the application layer above. In [20], Sundarajan et al. showed a viable solution to use Network Coding in conjunction with TCP. However, their transport layer approach requires kernel changes, lacking interoperability, resulting in a large update procedure. Our approach does not require fundamental changes and can be deployed by a simple application update, easing the deployment in existing networks.

B. *Network Coding*

Network Coding can be separated into intersession and intrasession Network Coding. Intersession Coding refers to coding across streams, mixing packets from different sources. It has proven to solve the bottleneck problem in a butterfly-network [1] and breaking the min-cut max-flow theorem for multicast communication [14]. In contrast, intrasession coding combines only packets from a single sender to increase loss tolerance by adding FEC. The here presented approach uses intrasession Network Coding, parameterized for constrained IEEE 802.15.4 devices. The following summary is based on our prior work [18].

The basic principle of Network Coding was first presented in [1]. There, Ahlswede et al. used a simple bitwise XOR operation to create an encoded packet from two original packets. The simple XOR approach was extended in [14] to Linear Network Coding (LNC), where a packet is treated as a vector over an extended binary Galois Field $GF(2^q)$. The field size is determined by the field exponent q. With this approach, data can be treated as a polynomial over the chosen field. For example, the 8-bit sequence 11001010 can be encoded 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.

A longer n bit data-sequence is represented by concatenating $l = \frac{n}{q}$ field elements. For example, a 128 byte message can be written as $l = \frac{128*8}{8} = 128$ concatenated $GF(2^8)$ elements. Using the notations from [20], an original packet p_1 can be formally described as:

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

This new representation shifts the data from the physical domain into an algebraic domain, allowing to do calculations across packets. Since Galois Fields are closed, the product or sum of two elements is another field-element, guaranteeing a uniform encoded packet-length. This allows to generate and transmit linear combinations of packets. For example, the coded packet q_1 can be created by $q_1 = \alpha_{11}p_1 + \alpha_{21}p_2$, as a linear combination of the original packets p_1, p_2, using the coding coefficients $\alpha_{21}, \alpha_{11} \in GF(2^q)$. Following [20], this encoding process for packets with a length of l elements can be written as a matrix multiplication:

$$\begin{pmatrix} q_{11} & q_{12} & \cdots & q_{1l} \\ q_{21} & q_{22} & \cdots & q_{2l} \end{pmatrix} = \begin{pmatrix} \alpha_{11} & \alpha_{21} \\ \alpha_{12} & \alpha_{22} \end{pmatrix} \cdot \begin{pmatrix} p_{11} & p_{12} & \cdots & p_{1l} \\ p_{21} & p_{22} & \cdots & p_{2l} \end{pmatrix} \quad (2)$$

Upon receiving a new linear independent packet, the receiver adds it to the decoder's storage, forming the decoding matrix D. If the rank $r(D)$ of this matrix equals the number of packets involved in the encoding process, called the generation size 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 $r(D)$ linear equations with g unknowns, which can be solved by using Gaussian elimination. Thus, the decoding condition can be written as: $r(D) \geq g$. If the decoding condition is satisfied, the inverted coefficient matrix C^{-1} can be used for decoding by inverting the encoding process:

$$\begin{pmatrix} p_{11} & p_{12} & \cdots & p_{1l} \\ p_{21} & p_{22} & \cdots & p_{2l} \end{pmatrix} = \begin{pmatrix} \alpha_{11} & \alpha_{21} \\ \alpha_{12} & \alpha_{22} \end{pmatrix}^{-1} \cdot \begin{pmatrix} q_{11} & q_{12} & \cdots & q_{1l} \\ q_{21} & q_{22} & \cdots & q_{2l} \end{pmatrix} \quad (3)$$

In [9] it is shown that the used coding coefficients α can be randomly chosen, while still achieving linear independency of the encoded packets with high probability. This improved coding scheme is called Random Linear Network Coding (RLNC). The implicit FEC ability of RLNC is given by the decoding condition. A receiver does not need to receive every single packet independently, but just has to obtain enough linear combinations to decode the accumulated matrix.

C. Related Work

An overview of existing Network Coding applications and their use in WSNs is given in [16]. Most existing applications, e.g., [11] or MIXIT [12], are routing solutions, which use overhearing on lower layers. This is not possible on the here targeted application layer. SenseCode [13] is a forwarding scheme explicit developed for sensor networks, but still lacks in interoperability. The closest related work is presented in [5], where a publish-subscribe system with Network Coding is introduced. While there are similarities, the shown approach differs from our work. It primarily investigates a gossiping algorithm to tackle timeliness in wide area networks. In our work, we focus on energy-efficiency for temporarily sleeping clients in WSNs with small packet sizes. Since a precise energy consumption measurement presents a problem for simulation, our evaluation is done in an experimental testbed, increasing the credibility of our results.

III. Architectural Design

In this section, we will present an efficient way to add Network Coding FEC to an MQTT-SN network. Our approach is a transparent application layer solution, which can be easily deployed to existing, running systems.

In our scenario, an end-user generally does not necessarily need the freshest data immediately, but does also not want to receive deprecated measurements. The nodes on the other hand try to turn off their radio module as long as possible to save battery power. Our approach uses a two phase duty cycle to achieve the needed trade-off between timeliness and energy efficiency. Most of the time, a node stays in a low power state, where it senses and stores the measurements but has its radio module turned off to save energy. In our scenario, a sensor node generates one data-packet every two minutes, but instead of transmitting it immediately, the packet is queued till the next transmission phase. To still satisfy timeliness constraints of the collected data, the node temporary enters a transmission phase. In this high power consumption phase, the node activates its radio module and transmits all prior collected data. Based on our experiences in an agricultural monitoring WSN and without loss of generality, we assume a one hour timeliness constrain.

A. Feedback Concept

The current MQTT-SN specification [19] does not contain or define a FEC scheme. To conform to the current MQTT-SN version 1.2, our here proclaimed architecture uses only existing publish-subscribe features. One alternate solution would be to extend the protocol by using a reserved message type *MsgType* [19, Chapter 5.2.2], e.g., *MsgType 0x11 = PUBLISH_FEC*, and specify its processing on reception. A similar approach would include a reserved *ProtocolId* [19, Chapter 5.3.8]. Both solutions are feasible in general, but

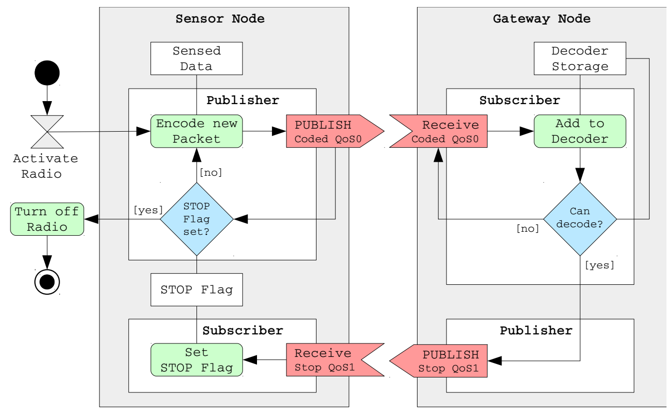

Fig. 3. Activity Diagram of one transmission phase using the presented architectural design.

differ from the official specification, leading to insufficient interoperability.

Figure 3 shows an activity diagram of our here presented architecture for one transmission phase of the publishing client. Instead of transmitting all queued data as reliable QoS 1 messages, 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. The core of our approach is to run a local subscriber on the gateway node and use an additional topic to exchange control messages between the gateway and the publishing client. After activating its radio, the sensor node encodes a new packet from the stored, sensed data and sends it as an unreliable PUBLISH message with QoS 0. When the gateway node subscriber receives such a coded packet, it feeds the encoded payload to its decoder. If the decoding-condition is satisfied, the gateway publisher sends out a "STOP"-control message, indicating that the sensed data was successfully transmitted. This single control message is send as a QoS 1 publication, using MQTT-SN's specified ARQ reliability. When the subscriber at the sensor node receives this control message, an internal flag is activated, stopping the publication of coded packets and allowing the node to enter its low power phase again.

B. *Seed-Based RLNC Scheme for Small Packets*

To successful decode a batch of received packets, the decoder needs to know, which coding coefficients were used in the encoding process of each packet. In a traditional RLNC implementation, the coding coefficients are appended as a vector to the packet. These additional bytes are called the coding overhead, which usually has a size of $\log_2(2^q) \cdot g$ bits [8], for a given Galois-Field size of 2^q and g involved original packets. A discussion of both parameters will be given later in this section. While transmitting this overhead is inefficient, the essential problem of Network Coding in WSNs is induced by the strict MTU constrains. The 802.15.4 standard is limited to 128 bytes on the physical layer, where half of these bytes could already be consumed for security or networking purposes, cf. [19, Chapter 2]. If the payload in our motivational deployment would contain an additional time stamp or another control parameter, it likely exceeds the IEEE 802.15.4 MTU limit.

The selected Galois-Field size determines the number of available coding coefficients that can be used to create encoded packets. Due to the efficiency of binary operations on modern CPUs, only extended binary Galois-Fields $GF(2^q)$ with field sizes of 2^q, $q \in \mathbb{N}$ are considered for RLNC. A larger exponent improves the probability to create linear independent packets, but also increases the complexity of the coding operations. Our proposed system uses $GF(2^8)$ with $q = 8$, since $GF(2^8)$ achieves good linear independence, while maintaining a reasonable coding complexity [8].

The generation size g represents the number of original packets from which the encoder can generate linear combinations. In our WSN scenario, a sensor node encodes over all queued packets when activating its radio interface. The queued packets correspond to the sensed data, gathered in the preceding low power phase. We have limited the low power phase to a maximum of one hour, due to the given scenario-specific timeliness constrains of the data (cf. Sec. III). Since a node queues one packet every two minutes, this leads to an accumulation of 30 packets when entering the transmission phase, resulting in a generation size of $g = 30$.

Since the coding overhead $(\log_2(2^q) \cdot g)$ is independent from the message length, most research in the Network Coding field assumes, the overhead is negligible, compared to the transmitted payload. While this assumption is reasonable for larger packets, it does not hold true when being limited by small MTUs, which is a common case in WSNs. In our scenario, the encoding is done over $GF(2^8)$, with a generation size of $g = 30$, resulting in an overhead of 30 bytes/packet. The payload in our real-world deployment has a size of only 29 bytes. Adding 30 additional bytes corresponds to a significant overhead of $\sim 103.45\%$.

To reduce the coding overhead, our solution uses a unique Network Coding scheme, called seed-based RLNC, which is presented in [4]. Using seed-based RLNC, the coding vector can be replaced with a single small number, if encoder and decoder can use this number as a seed to each initialize a random number generator (RNG), producing the same coding coefficients. A generic explanation of seed-based RLNC is given in [8]. In our system, the publisher selects the seed and initializes its RNG, and then generates the coefficients for encoding. The seed is then added to the head of the payload. Because the seed is significantly smaller than the usually added coding vector, an overhead reduction is achieved. For our scenario, we have chosen a one byte long seed, shrinking the overhead from 30 bytes down to 1 byte. On reception, the subscriber extracts the seed, initializes its RNG with it, then reproduces the coefficients to enable successful decoding.

When defining the seed size, a trade-off between overhead and loss resistance has to be made. A small size leads to a limited number of possible coefficient vectors. This restricts the creation of linear independent packets and could result in useless receptions, making the whole scheme inefficient. A large seed size, on the other hand, implies additional overhead. Our evaluation will show, that a one byte long seed is sufficient to achieve close to optimal Network Coding gains, while still only inducing a reasonable one byte small overhead.

An even more efficient way to get a pseudo-zero overhead would be to reuse the message identifier field *MsgId* of MQTT-SN. This two byte long field is included in every *PUBLISH* message header [19, Chapter 5.4.12]. For *QoS 0* messages, the field is set to $0x0000$. It is not further used and, thus, available for modification. Since the *MsgId* is already taken into account within the MQTT-SN header, this approach does not add additional overhead, as it reuses existing bytes. We still do not consider this solution as feasible for our scenario, since we focus on transparency and interoperability, restricting a modification of the MQTT-SN specification.

It has to be noted that there are some restrictions regarding recoding for seed-based RLNC. Due to our focus on the clients as endpoints and length limitations of this paper, we do not consider the effects of recoding in an MQTT-SN system any further.

IV. Implementation

To conduct the emulative testbed evaluation, we implemented the proposed architectual design for Linux clients. The implementation is based on *AsyncMQTT-SN* [22], which is an open source implementation of MQTT-SN. The presented Network Coding scheme was added to the application by creating a custom seed-based encoder and decoder, using the Kodo Network Coding library [17].

V. Evaluation

To evaluate the proposed solution, we conducted a three step approach. First, the collected real-world traces were analysed and a Gilbert-Elliott (GE) loss model was calibrated for each link of the real-world deployment. We decided to use a model and not to use the traces themselves. This allows to create a more replicable, controlled emulation setup. The model and its link-specific parameterization were then used to emulate bursty packet loss in an experimental testbed under controllable lab conditions. This testbed setup was also necessary, since the motivational real-world deployment had been removed after the vegetation period. In this second step, the interface uptimes were measured for each emulated link in order to evaluate the Network Coding gains against the MQTT-SN baseline approach. In the third evaluation step, precise power consumption measurements are presented. These measurements include the additional power consumption of the Network Coding scheme, which is induced by the increased CPU load. We will evaluate if the additional Network Coding operations have a significant effect on the overall power consumption.

A. Trace Analysis and Loss Model Calibration

The impact of the proposed MQTT-SN extension strongly depends on the quality of the link between both communication entities and the packet error ratio (PER), respectively. Hence, it is crucial to choose an adequate loss model if simulations or emulations are conducted for evaluating this impact. Moreover, credible evaluation demands a suitable calibration of such a model which commonly is scenario-specific and can be performed by analyzing real-world trace files. Since we focus on a real-world scenario and practical measurements of evaluation metrics in this paper, we decided to use an emulative evaluation approach in an experimental testbed based on *NetEm* [21], a widespread traffic control tool for Linux.

TABLE I
Link-specific parameterization of the Gilbert-Elliott loss model.

Link	PER in %	p in %	r in %	1-k in %	1-h in %	PER$_{GE}$ in %
$A_1 \to G$	22.03	0.814	2.749	0.295	88.594	19.42
$A_2 \to G$	20.29	0.960	3.485	0.347	85.145	18.57
$A_3 \to G$	18.01	0.782	3.595	0.250	86.240	13.80
$A_4 \to G$	15.61	0.955	4.696	0.286	81.788	13.38
$B \to G$	21.19	0.872	3.157	0.084	87.122	18.78

As a sound and comprehensive loss model selection for our scenario is out of scope of this paper, a classical Markov chain based loss model is used that is already included in NetEm, namely the GE model (cf. [7]). Furthermore, we made the assumption that link qualities are symmetric. The reason is that we observed a similar PER for both direction of links in our deployment since we use a common sensor node attached to a fully-equipped router as gateway device. Hence, for the calibration of the selected loss model, we use parameters obtained by links from the sensor nodes $A_1, .., A_4$ and B to their gateway G (cf. Fig. 1). The model is then applied for both link directions. The corresponding traces gathered during the long-term WSN deployment (74 days) contain a large number of the successfully delivered packets from each node to the gateway. Unsuccessful delivery attempts were also marked. Thus, with a number of roughly 23,500 packets per link, they serve as an appropriate and broad base for the calibration. The 29 byte payload of these packets consists of a unique source address, a sequence number (SN), and multiple sensor readings. The WSN application periodically senses and transmits this data with a fixed frequency of two minutes in daytime, using a IEEE 802.15.4 compliant CC2420 radio transceiver with maximum TX power at 2.4 GHz. We not only determined the PER of each individual link, but also analyzed the packet loss burstiness according to [6]. As g_min, the so-called gap threshold, a value of 4 packets is considered as reasonable for the burst classification of our trace files.

Based on this classification, we then determine relevant parameters for the GE model for all links in our deployment. These parameters are listed in Table I and emphasize the challenging connectivity in suchlike scenarios. For further calibration details, please refer to [7].

Finally, we evaluated the quality of our model and its calibration by comparing the average PERs as well as the caused burstiness. For the former, we conducted 50,000 ping mea-

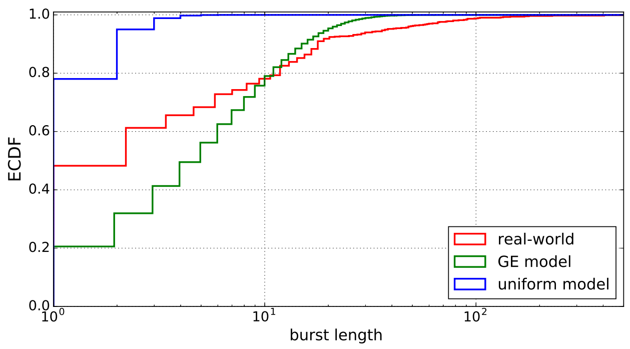

Fig. 4. Comparison of error burstiness of a real-world trace (link $A_1 \rightarrow G$) and of corresponding traces synthetically generated with a uniform distribution model and using the GE model with link-specific parameterization.

surements for each link, configured with NetEm and the corresponding GE model. The resulting PER is denoted as PER_{GE} and included in Table I. As can be seen, both PERs are roughly comparable. Figure 4 shows the burstiness of the real-world traces in comparison to traces that were synthetically generated by our model. The presented empirical cumulative distribution function (ECDF) plot shows exemplarily the link from sensor A_1 to the gateway G. The other links achieve very similar results. For demonstration purposes, we also included a second synthetic trace, generated by a conventional uniform distribution model with the corresponding average PER. The figure shows the benefit of the GE model since the simple uniform distribution model is obviously not representative for the real-world traces. The figure also confirms that the burstiness of both, the real-world and the GE model trace, is reasonable comparable as well, even though there is room for improvement. Therefore, a more complex model would be required. Yet, such a model is not supported by NetEm and the quality of the GE is sufficient for our purposes.

B. Testbed Setup

The constructed testbed consists of three different hardware platforms, representing the heterogeneous devices within the MQTT-SN topology of our motivational scenario. While the broker runs on a common notebook, a Raspberry Pi 3 model B was used as Gateway. The constrained in-field nodes are represented by a Raspberry Pi Zero with an attached TP-Link TL-WN722N WiFi dongle. The evaluated system uses UDP sockets, but the presented results are similar in IEEE 802.15.4 networks. We are aware, that the Pi Zero is on the upper spectrum of sensor nodes, regarding processing capability and power consumption. Yet, the shown findings can easily be transferred to other sensor node platforms, e.g. *Waspmote* [15]. Each link between a client and the gateway was emulated using *NetEm* [21] by applying the corresponding Gilbert-Elliot model, which was calculated from the corresponding real-world trace.

C. Measured Radio Uptime

In contrast to the simultaneously measured energy consumption, an evaluation of the radio uptime yields more transferable results. By using the measured uptimes and the nominally power consumption of a different sensor node platform, we can roughly predict the possible gains of our approach for other setups.

Figure 5 presents the measured interface uptimes for every emulated link. The MQTT-SN baseline approach is shown side-by-side to the novel Network Coding solution. Each boxplot involves 100 replications. The y-axis has a logarithmic scale. In the experiment, the client publishes once per second with a maximum of one in-flight message. This limit is used with respect to congestion and fairness in networks with constrained bandwidth. A duration of one second is also the shortest possible sending interval in the *AsyncMQTT-SN* client implementation. In the best practice section of the MQTT-SN specification, the recommended retransmission timeout is 10–15 s [19, Chapter 7.2]. While such a long timeout would amplify the benefits of our approach, we do not consider it as reasonable in our scenario. Thus, a lower retransmission timeout of 2 s is used. It has to be noted that the selected parameters have a large influence on the measurements and should be closer evaluated for other applications.

As shown in Figure 5, the Network Coding solution results in a shorter mean interface uptime for every emulated link than the MQTT-SN baseline approach. This reduced duration is achieved, because the Network Coding scheme uses unreliable, acknowledge-less QoS 0 publications. In the baseline scheme, each QoS 1 publication triggers a retransmission timeout, if either the publication message itself is lost or if the gateway's acknowledgement fails. For a symmetric link, this results in an average success probability of $(1 - p^2)$ for each baseline publication, where p is the link's average PER. The coded transmissions are sent with QoS 0 and, thus, do not need to be acknowledged. Except for the final, reliable control message, the average Network Coding success probability is $(1 - p)$ for

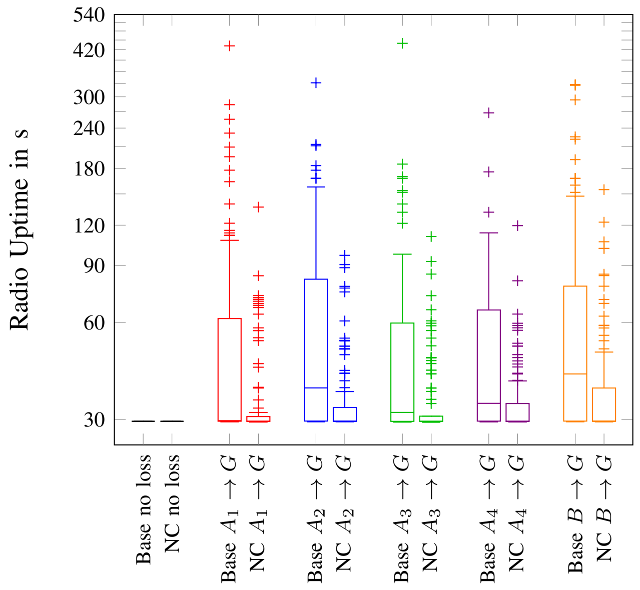

Fig. 5. Testbed measurement of the radio uptime for both transmission approaches, with each link emulated using the corresponding GE-model.

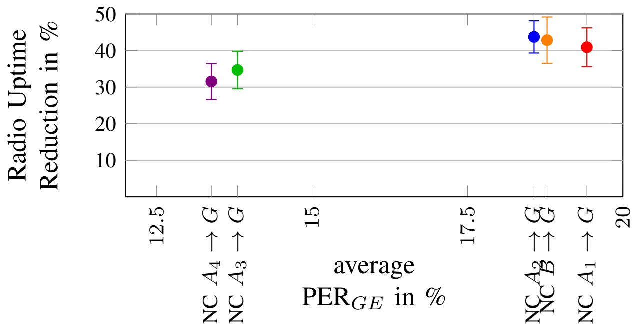

Fig. 6. Radio uptime duration reduction in % for each link, plotted ascending to the corresponding GE-model's average loss rate.

each publication, given optimal linear independency, which we will verify in the following Section V-D. This linear, instead of quadratic influence of the PER is a known benefit when using Network Coding FEC for reliable data transfer. The higher a link's PER, the more does the corresponding client profit from FEC. This is shown in Figure 6. A closer examination of this effect for increasing loss rates was done in [18]. For our testbed measurements, the system's overall average uptime reduction is 38.76 %.

D. Linear Independency Simulation

The linear independency of the encoded packets is a central criteria, when evaluating a Network Coding scheme. If a received packet is not independent from all already accumulated packets, it does not increase the decoder's rank. The packet will be discarded by the receiver, since it does not yield new useful information. In [9], it was proven, that encoded packets with randomly chosen coefficients are independent with high probability, for a sufficient large field size and generation size. In our setup, both selected parameters are large enough according to [9], since we code over $GF(2^8)$, using a generation size of $g = 30$

For a seed-based Network coding scheme, a linear independency evaluation has an additional, extraordinary priority, since the number of unique coding coefficient vectors is limited to the seed size. Our selected seed has a size of 1 byte, thus, 256 unique RNG runs can be initialized, leading to 256 different coefficient vectors, each inheriting RLNCs high probability of linear independency. If two packets would be received, that were encoded using the same seed, the later would be marked as linear dependent and rejected. This does not happen in standard RLNC, but is a unique, characteristic problem for seed-based schemes. To evaluate how often the problem occurs, we conducted an additional simulation, using longer, synthetic traces from all five calculated loss models. We noted when a received packet did not increase the decoder's rank. Out of over 40,000 received packets, only 4 of them were not linear independent. These packets will be treated by the receiver, as if the packets were lost. Compared to the channel-inducted packet loss in our scenario of 13.38 % to 19.42 % (cf. Tab. I), only 0.01 % of the received packets were rejected due to linear dependency. The influence of this effect is negligibly close to zero. While this empirical study is not a hard analytic proof, it verifies our selected seed size of 1 byte as reasonable for the given scenario.

E. Energy Consumption Measurements

To examine the hypothesis, that the additional coding operations on the CPU would lead to a higher power consumption, the testbed client's energy consumption in the transmitting phase was measured during the conducted uptime experiment. The electric current in Ampere was measured with a Fluke 289 True-RMS industrial multimeter, sampled with a polling rate of 1/s. The used power supply has a nominal voltage of 5 V. The client's power in Watt, calculated using the average current, is presented for each emulated link and transmission approach in Table II. The presented measurements do not confirm the hypothesis, that the additional coding operations on the CPU result in a higher power consumption. There is no significant difference or tendency regarding the power consumption between both schemes. It has to be noted, that the used Raspberry Pi Zero features a 1 GHz ARM1176JZF-S CPU, which does not reach its limit as fast as weaker CPUs in constrained sensor nodes. Since the overall average consumption is also higher, the additional CPU usage could be masked by side-effects, leaving room for future, dedicated research on this topic.

F. Scenario-specific Power Prediction

By combining the energy consumption measurements from Table II with the average radio uptimes, cf. Figure 5, we can predict the average daily energy consumption for the targeted motivational scenario. The final result in Figure 7 shows the average consumed transmission energy, accumulating over the course of one day. The sensor nodes in our motivational deployment measure the solar light intensity, thus, they are only active from 4:00 a.m. to 10:00 p.m. This interval is represented on the x-axis in Figure 7. At the beginning of every hour, each node activates its network interface and transmits the accumulated sensed data, then turns off the radio module again. Since the nodes consume in both compared approaches a similar amount of energy while transmitting, cf. Tab. II, the interface uptime is the deciding factor. Our proposed Network Coding solution enables the node to earlier turn off

TABLE II
MEASURED ENERGY CONSUMPTION

Link	Mode	V	mA	mW
$A_1 \rightarrow G$	*Base*	5.0	247 ± 0.25	1235
	NC	5.0	249 ± 0.32	1245
$A_2 \rightarrow G$	*Base*	5.0	249 ± 0.33	1245
	NC	5.0	249 ± 0.41	1245
$A_3 \rightarrow G$	*Base*	5.0	252 ± 0.27	1260
	NC	5.0	255 ± 0.26	1275
$A_4 \rightarrow G$	*Base*	5.0	246 ± 0.31	1230
	NC	5.0	252 ± 0.34	1260
$A_B \rightarrow G$	*Base*	5.0	251 ± 0.28	1255
	NC	5.0	249 ± 0.33	1245
All	*Sleeping*	5.0	84 ± 0.01	420

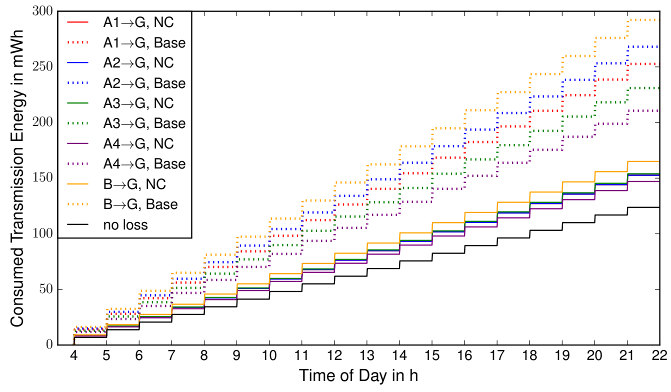

Fig. 7. Prediction of the average consumed radio interface energy consumption for one day in the motivational scenario

its radio, entering the low power state earlier. Thus, with every transmission phase, our approach saves energy, leading to the accumulating differences between the plotted same-colored graphs. Depending on the link quality, this results in a saving of 63.60 mWh for link $A_4 \rightarrow G$ up to 127.25 mWh per day for link $B \rightarrow G$. On average, our proposed Network Coding solution reduces the consumed energy in the transmission phases by 38.21 %, accumulating to an average daily saved 96.70 mWh for our motivational scenario.

While these savings are convincing, it has to be noted, that they are small, compared to the node's overall consumption. Each client runs on a Raspberry PI Zero, which already has a huge idle power of 420 mW (Tab. II). Since the client's radio is active for just a short duration every hour, the average saved 96.70 mWh only correspond to a 1.24 % reduction of the nodes' average overall power consumption of 7811 mWh. The benefits of our proposed solution are increased for sensor nodes, that have a larger gap in transmission and idle power consumption. For example, the widely used *Waspmote* platform [15] nominally consumes 185 to 320 mW while using its 5V XBee module and only 0.5 to 4.65 mW with its interface in a shallow sleep mode. For a client device with these parameters, our approach would roughly estimated save up to 146 mWh of the average daily consumed 486 mWh, which is an overall reduction of up to 30.04 %. These savings greatly increase the WSN's overall lifetime, which can make a costly, labor-intensive battery replacement obsolete to achieve a maintenance-free deployment period.

VI. Conclusion

The conducted testbed experiments have shown convincing results. The proposed architectural design can easily be implemented and transparently deployed to existing WSNs. By using the presented seed-based Network Coding scheme, the sensor nodes radio uptimes will be reduced to save energy, while still guaranteeing reliable data transfer. For the given loss rates in our scenario, the radio uptimes can be reduced by 38.24 % on average. We are looking forward to further evaluate our approach in oncoming field studies and further measure the energy savings for stronger constrained devices.

References

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

[2] A. Banks and R. Gupta (Eds.). (2014, October) MQTT version 3.1.1. OASIS Standard. [Online]. Available: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html

[3] 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*, vol. 128, pp. 149–159, 2016.

[4] C.-C. Chao, C.-C. Chou, and H.-Y. Wei, "Pseudo Random Network Coding Design for IEEE 802.16m Enhanced Multicast and Broadcast Service," in *Proc. of the 71st IEEE Vehicular Technology Conference (VTC Spring)*, 2010, pp. 1–5.

[5] C. Esposito, M. Platania, and R. Beraldi, "Reliable and timely event notification for publish/subscribe services over the internet," *IEEE/ACM Transactions on Networking*, vol. 22, no. 1, pp. 230–243, 2014.

[6] T. Friedman, R. Caceres, and A. Clark, "RTP control protocol extended reports (RTCP XR)," RFC 3611, IETF Network Working Group, Nov. 2003.

[7] G. Hasslinger and O. Hohlfeld, "The Gilbert-Elliott model for packet loss in real time services on the internet," in *Proc. of the 14th GI/ITG Conference on Measurement, Modelling and Evalutation of Computer and Communication Systems*, Dortmund, Germany, 2008, pp. 1–15.

[8] J. Heide, M. V. Pedersen, F. H. Fitzek, and M. Médard, "On code parameters and coding vector representation for practical RLNC," in *Proc. of IEEE Int. Conference on Communications (ICC)*, Kyoto, Japan, 2011, pp. 1–5.

[9] T. Ho, M. Médard, R. Koetter, D. R. Karger, M. Effros, J. Shi, and B. Leong, "A random linear network coding approach to multicast," *Information Theory, IEEE Transactions on*, vol. 52, no. 10, pp. 4413–4430, 2006.

[10] "ISO/IEC 20922:2016, Information technology — Message Queuing Telemetry Transport (MQTT) v3.1.1," International Organization for Standardization, Geneva, Switzerland, June 2016.

[11] D. Katabi, S. Katti, W. Hu, H. Rahul, and M. Medard, "On practical network coding for wireless environments," in *Proc. of the Int. Zurich Seminar on Communications*, Switzerland, 2006, pp. 84–85.

[12] S. Katti, D. Katabi, H. Balakrishnan, and M. Medard, "Symbol-level network coding for wireless mesh networks," in *ACM SIGCOMM Computer Communication Review*, vol. 38, 2008, pp. 401–412.

[13] L. Keller, E. Atsan, K. Argyraki, and C. Fragouli, "SenseCode: network coding for reliable sensor networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 9, no. 2, pp. 25–33, 2013.

[14] S.-Y. R. Li, R. W. Yeung, and N. Cai, "Linear network coding," *Information Theory, IEEE Transactions on*, vol. 49, no. 2, pp. 371–381, 2003.

[15] Libelium. (2017, April) Waspmote power programming guide. [Online]. Available: https://www.libelium.com/v11-files/documentation/waspmote/waspmote-power-programming_guide.pdf

[16] P. Ostovari, J. Wu, and A. Khreishah, "Network coding techniques for wireless and sensor networks," in *The Art of Wireless Sensor Networks*. Springer, 2014, pp. 129–162.

[17] M. V. Pedersen, J. Heide, and F. H. Fitzek, "Kodo: An open and research oriented network coding library," in *Proc. of the IFIP TC 6th Int. Conference on Networking (NETWORKING)*, Valencia, Spain, 2011, pp. 145–152.

[18] B. Schütz and N. Aschenbruck, "Adding a network coding extension to CoAP for large resource transfer," in *Proc. of the 41st IEEE Conference on Local Computer Networks (LCN)*, Dubai, UAE, 2016, pp. 715–722.

[19] A. Stanford-Clark and H. L. Truong. (2013, November) MQTT for sensor networks (MQTT-SN) protocol specification version 1.2. IBM. [Online]. Available: http://mqtt.org/new/wp-content/uploads/2009/06/MQTT-SN_spec_v1.2.pdf

[20] 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*, vol. 99, no. 3, pp. 490–512, 2011.

[21] The Linux Foundation. (2017, July) netem. [Online]. Available: http://www.linuxfoundation.org/collaborate/workgroups/networking/netem

[22] T. Yamaguchi. (2017, April) AsyncMQTT-SN.