Monitoring Bats in the Wild: On Using Erasure Codes for Energy-Efficient Wireless Sensor Networks

Falko Dressler, University of Paderborn
Margt Mutschlechner, University of Innsbruck
Bijun Li, TU Braunschweig
Rüdiger Kapitza, TU Braunschweig
Simon Ripperger, Museum für Naturkunde, Leibniz Institute for Evolution and Biodiversity Science
Christopher Eibel, FAU Erlangen-Nürnberg
Benedict Herzog, FAU Erlangen-Nürnberg
Timo Hönig, FAU Erlangen-Nürnberg
Wolfgang Schröder-Preikschat, FAU Erlangen-Nürnberg

We explore the advantages of using Erasure Codes (ECs) in a very challenging sensor networking scenario, namely monitoring and tracking bats in the wild. The mobile bat nodes collect contact information that needs to be transmitted to stationary base stations whenever being in communication range. We are particularly interested in improving the overall communication reliability of the wireless communication. The mobile nodes are capable to store a few 100 kB of data and to exchange contact information in aggregated form. Due to the continuous flight of the bats and the forest environment, the wireless channel quality varies quickly and, thus, the communication is in general assumed to be highly unreliable. Given the very strict energy constraints of the mobile node and the inherently asymmetric channels, conventional techniques such as full data replication or Automatic Repeat Request to improve the communication reliability are prohibitive. In this work, we investigate the trade-off between reliability achieved and the cost in form of additional transmissions, i.e., the additional energy costs. Our energy measurements on a real platform combined with larger scale simulation of the wireless communication clearly indicate the advantages of using ECs in our scenario. The results are also applicable in other configurations when unreliable communication channels meet tight energy budgets.

Categories and Subject Descriptors: C.2.2 [**Computer-Communication Networks**]: Network Protocols

General Terms: Design, Algorithms, Performance

Additional Key Words and Phrases: Wireless sensor networks, erasure codes, forward error correction, reliable communication, energy efficiency

ACM Reference Format:
Falko Dressler, Margit Mutschlechner, Bijun Li, Rüdiger Kapitza, Simon Ripperger, Christopher Eibel, Benedict Herzog, Timo Hönig, and Wolfgang Schröder-Preikschat, 2016. Monitoring Bats in the Wild: On Using Erasure Codes for Energy-Efficient Wireless Sensor Networks. *ACM Trans. Sensor Netw.* 12, 1, Article 7 (February 2016), 30 pages.
DOI: http://dx.doi.org/10.1145/10.1145/2875426

This work has been supported by the German Research Foundation (DFG) under grant no. FOR 1508.
Author's addresses: Falko Dressler, Dept. of Computer Science, University of Paderborn; Margit Mutschlechner, Institute of Computer Science, University of Innsbruck; Bijun Li and Rüdiger Kapitza, Dept. of Computer Science, TU Braunschweig; Simon Ripperger, Museum für Naturkunde, Leibniz Institute for Evolution and Biodiversity Science; Christopher Eibel, Benedict Herzog, Timo Hönig, and Wolfgang Schröder-Preikschat, Dept. of Computer Science, FAU Erlangen-Nürnberg.

DOI: http://dx.doi.org/10.1145/10.1145/2875426

1. INTRODUCTION

Since the early days of sensor network research, sensor networking technology has frequently been proposed for the use in wildlife monitoring. Among a few other applications, this has been one of the most successful examples making its way into real world usage. Sensor networking based wildlife monitoring provides more sophisticated methods for biologists to study a specific species, in terms of gathering a huge amount of data by long-term observations. The first projects relied on typical sensor platforms as used in academic research labs, e.g., the Great Duck Island project [Mainwaring et al. 2002], or on special hardware that is even robust enough to be carried by larger animals, e.g., the ZebraNet project [Juang et al. 2002]. Besides the manpower and other resources saved by employing sensor nodes instead of human beings, they succeeded in maintaining a reliable system with a very high data collection rate.

In more recent activities, heterogeneous sensor nodes have been used for tracking generic animals and endangered species such as Iberian lynx in the surrounding area of wildlife passages, which was built to establish safe ways for animals to cross transportation infrastructures [Garcia-Sanchez et al. 2010]. This system allows target identification through the use of video sensors connected to strategically deployed nodes. Recently, a sensor networking system has been presented used for monitoring rats [Link et al. 2010]. Furthermore, the development of the "Encounternet" system allowed for the successful mapping of social networks in wild New Caledonian crows [Rutz et al. 2012]. Conceptually similar to our system, nodes are attached to rats or crows to observe their contacts and routes.

From these successful approaches to wildlife monitoring using sensor networks, we learned about hardware design issues, network management, and data collection techniques. In the new BATS[1] project on monitoring the group dynamics of bats in their natural habitat, we go one step further and investigate potentials of ultra-low power sensor systems carried by the bats to monitor contacts or *encounters* between individuals and to track their routes at high spatial and temporal resolution [Dressler et al. 2016]. The aim of the project is to support biologists with their study on bats, a strictly protected animal taxon in the European Union, to track their living habitats and social behaviors. Mouse-eared bats (*Myotis myotis*) are the main study target [Arlettaz 1996; Rudolph et al. 2009]. The key challenge is that the animals with a minimal weight of about 20 g can carry sensors of at most 2 g for node plus battery, which strongly limits the available energy budget as well as the computational power and storage capabilities. Comparable sensors published in the literature typically weigh more than 100 g, with the Encounter tags being outstanding with a weight of only 20 g.

The scenario employs mobile nodes which are situated on bats and base nodes on the ground, as shown in Figure 1. In order to document social interactions among bats, all tagged individuals continuously exchange contact information. Data completeness is essential for the construction and proper analysis of social networks. However, the observed bats only appear in the communication range of a base station on an irregular basis. If in communication range, they are supposed to upload all contact information. In the following, we will refer to this data as *chunks* representing sets of contact information collected by a mobile node on the bat.

Unfortunately, the channel quality may vary quickly due to the continuous movements of bats and the heterogeneous environment, thus, the communication is in general assumed to be highly unreliable. Conventional reliability-improving approaches such as full data replication or on-demand retransmission are too expensive or even not possible due to very strict energy constraints and asymmetric channels. This is of course in

[1]Dynamically adaptive applications for bat localization using embedded communicating sensor systems, http://www.for-bats.org/

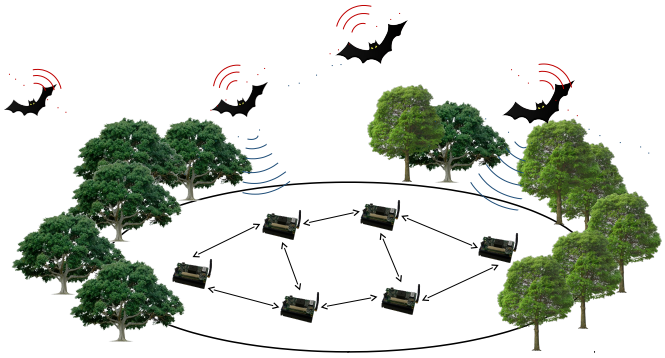


Fig. 1: The BATS deployment scenario.

conflict with the objective to improve the overall sensor network lifetime [Dietrich and Dressler 2009].

1.1. Communication Architecture

Our bats monitoring scenario is based on mobile nodes attached to individual bats and a stationary ground network using base stations deployed within the hunting area. The ground network is responsible for flight tracking using on the fly localization techniques as well as for collecting encounter information from the bat nodes.

All mobile nodes exchange beacon messages to collect encounter information. These encounters are uploaded to the base stations whenever the mobile node gets into contact with such a ground station. We obviously have two options: using a distributed ground network or storing the information at a base station located at the roost where the bats stay during the day time. The very limited storage capacities on our ultra-low power target platform only allow to keep a very limited number of encounters in memory. In addition, live experiments capturing the encounters of bats are only possible when triggering a upload as early as possible. We therefore decided to follow a fully distributed approach by uploading the encounter information as soon as the mobile node meets a ground station.

The overall communication architecture is shown in Figure 2. Using a multi-stage wake-up receiver, the downlink is controlled by stationary base nodes in the ground network. Whenever a bat node gets into the communication range of a ground node, its wake-up signal initiates the upload of contact information, i.e., all available data chunks in form of multiple radio packets.

As mentioned before, the quality of the communication channel is inherently varying. We study the use of Erasure Codes (ECs) in this scenario for error control and forward error correction in order to improve the overall reliability of the channel. Each message is assumed to be protected by a CRC, which allows to identify bit errors. The alternative would be to use coding directly at the physical layer using ECs or Low-Density Parity-Checks (LDPCs) codes [Gallager 1962; Pfister and Siegel 2008]. We employ simple retransmissions as well as ECs for this upload to investigate the advantages of each solution.

1.2. Contributions

In this paper, we investigate the use of ECs to improve the communication reliability between the mobile nodes and the ground network. In earlier work, we already explored

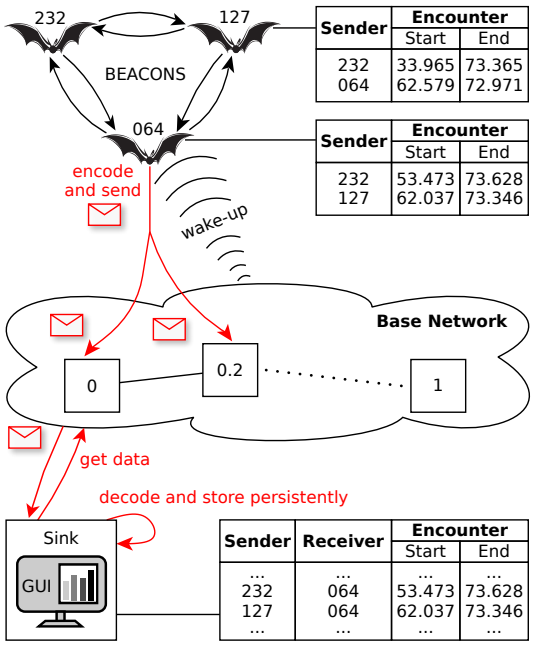


Fig. 2: Communication architecture.

the basic idea of this concept [Mutschlechner et al. 2014b; Mutschlechner et al. 2014a]. Compared to the simplistic approach to send chunk replica together with the original data chunk, ECs offer a better performance with reduced costs. Likewise, ECs show a better efficiency than on-demand chunk retransmissions realized by acknowledging successfully transmitted chunks. We carefully investigate the performance of three types of ECs in terms of reliability improvement via simulations, since the target hardware is still under development at this moment. One of the most critical characteristics of the used simulation model is the mobility pattern of the bats. We based this model on empirical data on free ranging bats provided by biologists.

Extending our previous work in [Mutschlechner et al. 2014b], we now present a more accurate mobility model for the hunting bats, which has been validated using empirical data from published field observations. We also updated the communication model and compared the use of ECs to simple Automatic Repeat Requests (ARQs). Furthermore, we specifically assessed the computational feasibility of using ECs on the target micro controller.

According to our findings, we can report several advantages of using ECs, which have not yet been considered in other sensor data upload applications. It is clear that the redundancy introduced by using ECs or data replication will increase energy consumption with the number of additional chunks. Simulation results show that with the same or even less overhead compared to classical data replication and on-demand retransmission, ECs can provide a higher degree of reliability in our specific application scenario.

Our key contributions can be summarized as follows:

— We introduce the use of ECs in the field of low-energy sensor tracking systems for wildlife monitoring (Section 3).

— We developed and carefully validated a new mobility model able to characterize the flight paths of bats during their hunting process (Section 4). This model allows to very accurately study wireless communication in simulation and to derive typical contact patterns and times between individuals.
— We developed a novel communication protocol based on erasure codes for disseminating contact information from the mobile bat nodes to stationary ground nodes (Section 5).
— We investigated both the energy footprint and the computational overhead of ECs on a real hardware platform (Section 6.1).
— Furthermore, we performed an extensive set of simulation experiments to explore the potential of ECs compared to other solutions (Section 6.2).

2. RELATED WORK

Sensor networks have successfully been used for wildlife and habitat monitoring for a long time already. First projects have been Great Duck Island [Mainwaring et al. 2002] and ZebraNet [Juang et al. 2002]. In these activities, sensor networking technology has been explored primarily to answer questions such as how to transmit sensor readings to a connected base station or a gateway, or how long does this network operate given the prototype sensor nodes and off-the-shelf batteries. More recently, sensor technology in combination with GPS receivers has been applied to track movement and behavior of individual animals [Garcia-Sanchez et al. 2010; Anthony et al. 2012; Sommer et al. 2014]. Furthermore, multiple sensor streams allowed for event type dissociation in order to reduce power consumption of GPS receivers and to extend operation times [Jurdak et al. 2013]. Also, the technological advances enabled new generations of sensor nodes that can be used to track much smaller animals such as rats [Link et al. 2010]. Wireless digital transceiver technology rendered even the automated mapping of social networks in wild birds possible [Rutz et al. 2012]. We go one step further and develop a sensor platform that allows for both localization and documentation of encounters weighing 2 g including a 1 g battery. Thus, known communication principles need to be reconsidered especially with respect to the reliability and the resulting lifetime [Dietrich and Dressler 2009]. In particular, we explore ECs as a well known technique for forward error correction to reduce the number of repeated message transmissions considering the underlying unreliable wireless communication channels.

Erasure codes are widely employed to improve the reliability in wireless transmissions in general as well as in wireless sensor networks [Kim et al. 2004]. The usage of EC techniques for wireless transmissions without a feedback channel has been investigated in [Berger et al. 2008]. Here, the optimal trade-off between error-correction coding within packets and erasure-correction coding across packets has been determined. The authors show that the trade-off depends on both the fading statistics and the average Signal to Noise Ratio (SNR) of the wireless channel, where for severe fading channels the trade-off leans towards more redundancy across packets and less redundancy within each packet. Hence, since we are facing a highly unreliable channel, the application of ECs adding redundancy across packets is appropriate.

In [Angelopoulos et al. 2013], Random Linear Network Coding (RLNC) has been proposed as a packet-level EC in combination with intra-packet error correction at the physical layer for low data rate indoor Wireless Sensor Networks (WSNs). The results indicate that RLNC at a code rate of $r = \frac{4}{8}$ provides an SNR improvement of 3.4 dB and a gain of 5.6 dB when combined with intra-packet error correction.

Also several studies to compare ECs with traditional reliability enhancing approaches such as data replication and ARQ have been conducted. A cross-layer methodology for analyzing error control schemes in WSNs has been proposed in [Vuran and Akyildiz 2009]. The analysis includes a comprehensive comparison of ARQ and several Forward

Error Correction (FEC) codes. The results presented outline that FEC codes are well suited as reliability improvement technique in delay-sensitive WSNs since energy consumption and the end-to-end latency is reduced. Furthermore, it has been shown that this improvement can be exploited by employing transmit power control and hop length extension.

To the best of our knowledge, there exists no study on the feasibility of ECs for scenarios with spontaneous connectivity such as the scenario we are investigating with its specific channel properties.

3. ERASURE CODES

An Erasure Code (EC) is a FEC code for the erasure channel that enhances data transmission reliability by introducing redundancy, however, without the overhead of strict replication. In the presented usage scenario, erasures take place on a per-packet basis, hence, ECs are used to introduce inter-packet redundancy.

ECs consist of an encoding and a decoding algorithm. The former one extends a group of k packets to n packets by generating $m = n - k$ redundant packets, where $k < n$. Each subset of the n packets containing at least k' packets is sufficient to successfully decode the original data, where $k \leq k'$. The code rate $r = \frac{k}{n}$ describes the overhead in terms of redundant packets.

There exist various kinds of ECs. To identify the most suitable EC, we accomplished a study to determine feasible candidates. We then evaluated the most promising ECs with the help of simulations as well as energy measurements on a real hardware platform.

3.1. EC Selection

In general, ECs can be divided into optimal and nearly-optimal ECs. Optimal ECs, such as Reed-Solomon (RS) codes [Reed and Solomon 1960], have the property that any k out of n packets are sufficient to successfully decode the original data, i.e., $k' = k$. Nearly-optimal ECs, for example Tornado codes [Luby et al. 2001], introduce a slight overhead such that $k' = (1 + \epsilon) * k$ packets are required to decode the data successfully, where $\epsilon > 0$, hence, $k' > k$. However, the encoding and decoding algorithms are less expensive. They have a linear complexity with respect to n, whereas optimal ECs can have up to quadratic coding complexity for large n.

In recent years rateless ECs, such as Luby Transfom (LT) codes [Luby 2002] and Rapid Tornado (RAPTOR) codes [Shokrollahi 2006], evolved. These are a special kind of nearly-optimal ECs where the encoding algorithm generates a potentially infinite amount of redundant data without having a fixed code rate. The main advantage emerges in a scenario with multiple receivers where a feedback channel is present. The encoding entity generates and transmits redundant data up until obtaining a notification about the successful decoding from all receivers. If a receiver holds an insufficient amount of packets, i.e., the amount of received packets is smaller than k', it must obtain not yet received packets in order to be able to decode successfully. Since these might be distinct packets for each receiver, the encoding entity might have to retransmit multiple packets individually for each receiver when using fixed-rate ECs. The encoding algorithm of rateless ECs, however, produces an infinite amount of redundant data, hence, transmitting a newly generated redundant packet is suitable for each receiver. Thus, the amount of transmissions is reduced.

According to our study, which has been confirmed also in [Angelopoulos et al. 2013], optimal ECs are most suitable for the presented scenario. In order to use rateless nearly-optimal ECs effectively, a feedback channel is needed. This, however, is not given in our scenario due to the high mobility of the nodes (cf. Section 4.2). Moreover, the encoding entity is highly energy constrained, hence sending an unlimited amount of

redundant data is clearly not feasible. Therefore, rateless ECs have been excluded from the simulation.

Regarding nearly-optimal codes in general, the overhead introduced by ϵ is a major drawback in our scenario. The necessary value for ϵ increases as the amount of original data k decreases. Hence, the amount of data required for decoding k' is growing if k decreases. To obtain a low overhead, k is supposed to have a large value, however, this is not achievable by the highly energy-constrained mobile nodes responsible for the encoding. Our mobile nodes have a restricted amount of storage (a few 100 kB) due to the highly limited node weight and size. Furthermore, mobile nodes may have only infrequent contact to the base network, hence, waiting until enough data is gathered to get a large value for k might result in forfeiting the rare communication possibilities.

In contrast, we have the drawback of optimal ECs exhibiting a higher coding complexity for large n. However, since the amount of original and redundant data is supposed to be very small, this is negligibly low for our scenario. Therefore, we mainly focus on optimal ECs, using a selected nearly-optimal EC for comparison purposes. In particular, we rely on the following existing open source implementations:

— Cauchy: an RS code based on a Cauchy matrix, developed by Michael Luby [Blömer et al. 1995]
— Vandermonde: an RS code based on a Vandermonde matrix, developed by Luigi Rizzo [Rizzo 1997]
— Tornado: a nearly-optimal EC, developed by Michael Noisternig [Noisternig 2004]

Each one of the three implementations has been evaluated with four different code rates: $r = \{\frac{4}{5}, \frac{4}{6}, \frac{4}{7}, \frac{4}{8}\}$. As a baseline experiment, we simulated the scenario with no reliability improvement, i.e., data is sent without encoding. Furthermore, for comparison reasons we integrated the full replication idea, i.e., data is sent together with an exact replica to increase reliability, and on-demand retransmissions, i.e., data is resent until receiving an acknowledgment indicating a successful reception of the data.

3.2. Coding Algorithms

The significant difference between the various ECs are the mathematical operands and operations that are used during the encoding and decoding process. The two RS codes Cauchy and Vandermonde apply the same kind of mathematical operation in their encoding and decoding algorithms, however, they work on different kinds of matrices, whereas Tornado varies significantly in the algorithm, i.e., the applied operations itself.

RS codes are cyclic block codes that split the original data x into k equally sized blocks $x_1 \dots x_k$. These blocks are considered as the coefficients of a polynomial over a finite field F:

$$P_x(c) = \sum_{i=1}^{k} x_i c^{i-1} \tag{1}$$

The encoding algorithm extrapolates it at n distinct sampling points $P_x(c_1) \dots P_x(c_n)$, where the first k points $P_x(c_1) \dots P_x(c_k)$ correspond to the original blocks (thus, successfully receiving the first k blocks corresponds to an error-free transmission of the original messages). This encoding function is a linear mapping and can be realized as $x \to x \times A$, where A is a $k \times n$ generator matrix with elements from F. Therein lies the difference between the two chosen RS codes since they use a Cauchy and a Vandermonde matrix, respectively. The decoding algorithm inverts the encoding by interpolating over some of the values of $P_x(c_1) \dots P_x(c_n)$, where at least k out of the n sampled points are needed to recover the original blocks. The encoding and decoding algorithms have a complexity of $\mathcal{O}(n \cdot \log n)$ and $\mathcal{O}(n^2)$, respectively, where the computational effort of the decoding

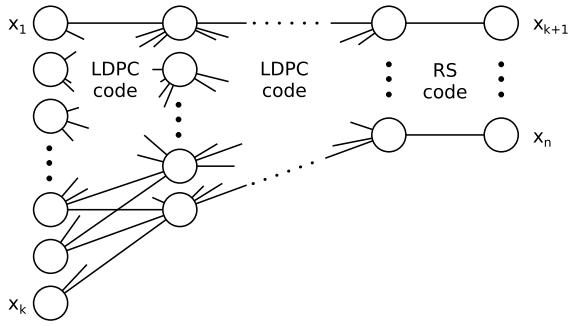


Fig. 3: Encoding process of Tornado.

algorithm is not crucial since only the encoding is performed by the energy-constrained nodes.

In contrast, the encoding and decoding algorithms of Tornado are based on a bipartite irregular graph [Luby et al. 2001]. The original data is split again into k equally sized blocks $x_1 \dots x_k$, each one represented by one node in the graph. The algorithm is realized in multiple levels visualized in Figure 3.

Each level except the last one performs a LDPC code by combining multiple nodes of the graph with an inexpensive XOR operation to generate redundant data. For each level the nodes themselves can be chosen randomly without repetition, however, the total amount is given by a specific distribution for each level. This operation is performed recursively until the final level encodes the nodes with an RS code. Since each level reduces the number of nodes, the complex operations during the final level have to be performed on a much smaller subset.

Decoding reverses the encoding algorithm by executing the RS decoding algorithm before applying the XOR operations starting from the second to last level. Both the encoded and decoded algorithms have a complexity of $\mathcal{O}(n \cdot \ln \frac{1}{\epsilon})$, where ϵ is a positive constant representing the overhead needed for decoding.

4. MOBILITY MODEL AND CONTACT PATTERN

The mobility model of mobile nodes has a high impact on the results of the simulation. It influences both the quality of the communication channel as well as the duration of the communication between mobile and base nodes. In the real deployment scenario each mobile node corresponds to a bat, and the simulated area resembles the foraging patch. Therefore, in order to have realistic results, the mobility model of mobile nodes must resemble the flying behavior of the observed species during foraging in the most realistic way.

4.1. Simulation Model

The simulation scenario is depicted in Figure 4, which reflects the envisioned deployment scenario. It has a total size of 5 km$\times$5 km and consists of two area types: the hunting grounds and the areas in between. For the sake of simplicity all hunting grounds have rectangular shapes in the simulation, displayed as grey areas in Figure 4. In the simulation there exist eight different hunting grounds with sizes ranging from 10 ha–50 ha as described in [Rudolph et al. 2009; Audet 1990; Arlettaz 1996]. Moreover, the daytime roost is a denoted point within the simulated area, visualized as black area in Figure 4.

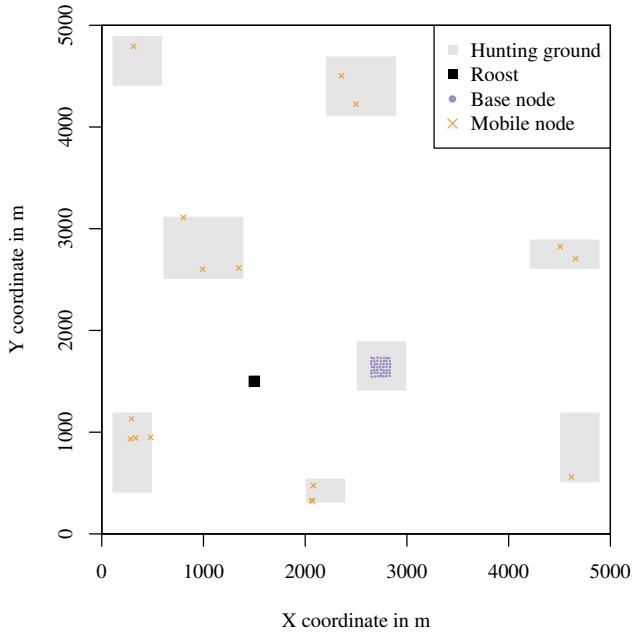


Fig. 4: A two-dimensional overview of the simulated scenario.

The scenario consists of two node types: mobile nodes situated on bats and base nodes laid out in the habitat to form a stationary backbone network, visualized in Figure 4 as violet rectangles and orange dots, respectively. In total 16 mobile nodes and a base network of 49 nodes are deployed. The base nodes are arranged in form of a grid-shaped but not fully regular manner with a distance of 30 m between each other in one of the hunting areas where the bats are mainly located. Moreover, a central node to ease the gathering of statistical information is integrated.

All the simulations are based on the OMNeT++ simulation core [Varga 2001], a discrete and event-based simulation framework. In one run we simulated approximately 5.5 h, which reflects the activity period of a bat between exiting and reentering the roost according to [Rudolph et al. 2009].

4.2. Bat Mobility Model

Although the specific characteristics of bat movements are not completely known yet, various figures in terms of flight speed, flight height, and flight routes are known nowadays [Arlettaz 1996; Rudolph et al. 2009; Russo et al. 2007; Skiba 2003; Audet 1990]. Furthermore, differences to the flying behavior of birds can be found [Hedenström et al. 2009]. To the best of our knowledge there exists no mobility model for the specific flight of bats during foraging, however, there are various mobility models suitable as a basis.

We decided to adapt an existing mobility model to fit the special characteristics of bat flights. The Lévi flight model is well-known for describing the movements of free-living animals. It has been shown to resemble the flight of birds [Viswanathan et al. 1996]. It is based on the random waypoint model with a heavy-tailed probability distribution for the step length. Therefore, we decided to rely on the random waypoint model, which was extended by the foraging movement patterns of mouse-eared bats (*Myotis myotis*)

Movement Pattern	Speed	Height	Next Pattern(s)
P1 Move to hunting ground	30 km h^{-1}–50 km h^{-1}	5 m–15 m	*T2* upon arrival within hunting ground
T2 Change to *P2*	-	-	*P2* when target height is reached
P2 Search for prey	15 km h^{-1}–35 km h^{-1}	1 m–2 m	*P1* or *P3* randomly, *P7* after 5.5 h
P3 Hovering	0 km h^{-1}	1 m–2 m	*T4* after 0.5 s–1 s
T4 Change to *P4*	-	-	*P4* when target height is reached
P4 Prey capturing	0 km h^{-1}	0 m	*T2* or *T5* with 25 % and 75 %, respectively
T5 Change to *P5*	-	-	*P5* when target height is reached
P5 Prey consumption	15 km h^{-1}	5 m–10 m	*T2* or *T6* after 10 s–20 s with 25 % and 75 %, respectively
T6 Change to *P6*	-	-	*P6* when target height is reached
P6 Resting	0 km h^{-1}	3 m–5 m	*T2* after 1 min–130 min
P7 Return to roost	30 km h^{-1}–50 km h^{-1}	5 m–15 m	-

Table I: The various movement patterns of the mobility model.

described in [Arlettaz 1996; Rudolph et al. 2009; Russo et al. 2007; Skiba 2003; Audet 1990].

For most of the simulation parameters we found more or less exact figures and value ranges in the literature. However, for the remaining parameters where not enough profound data exists we assigned values in a most realistic range advised by biologists. In the following we describe the mobility model in detail and give references for the parameters.

During the simulation, a mobile node is in one of seven different movement patterns *P1* to *P7*. It switches between them in a well-defined manner to imitate the different behavioural characteristics of bats while foraging. Additionally there exist some transitional phases named *T2*, *T4*, *T5*, *T6*, and *T7* in order to switch to the speed and height of the next movement pattern. The various movement patterns and transitional phases are summarized in Table I and are explained in the following in more detail.

At simulation start each mobile node, i.e., the bat, is located at the same place, the roost. Within 0 s–60 s, a mobile node starts moving according to *P1* where it moves towards a random hunting ground at a height of 5 m–15 m and a speed of 30 km h^{-1}–50 km h^{-1} [Audet 1990; Arlettaz 1996]. When reaching a hunting ground, it changes its behaviour in order to simulate the search for prey. The mobile node changes to this next movement pattern by passing through transitional phase *T2*. It slows down to 15 km h^{-1}–35 km h^{-1} and lowers itself to a height of 1 m–2 m according to findings described in [Arlettaz 1996; Skiba 2003].

When the target height is reached, it switches to *P2* where it moves randomly throughout the hunting ground in between these height and speed limits. It stays in this phase until moving to a different hunting ground or catching prey. The former case is accomplished by switching back to *P1* after a random time between 2400 s–3000 s. This results in approximately eight visited hunting grounds per night, which correlates to [Audet 1990; Arlettaz 1996; Rudolph et al. 2009]. The prey catching is triggered randomly such that during one simulation a mobile node switches approximately 20 times to this phase, which can be motivated by the amount of food needed by the target bat.

Each night such a bat needs to feed approximately 10 % of its body weight. Hence, with a weight of roughly 25 g–30 g the bat has to capture 2.5 g–3 g of insects [Audet 1990]. Assuming their prey to weigh about 150 mg–300 mg a bat would have to catch 8–20 specimen per night. However, since indigestible parts like legs and shard were not considered in this calculation, we set the value slightly higher.

Since bats are agile and talented hunters, biologists assume a capture success rate of roughly 75 %, which results in average to 5 failed captures and 25 catches in total. In order to simulate the prey catching the mobile node first changes to *P3* where it hovers in the air for 0.5 s–1 s as described in [Russo et al. 2007]. Then, it passes through the transitional phase *T4* where it moves to the ground within a radius of 1 m. When reaching the ground the mobile nodes switches to *P4* and stays at the same position for 1 s–2 s to imitate the prey catching [Arlettaz 1996].

As motivated prior, the capture failed with a probability of 25 % and the mobile node changes directly back to *P2* [Arlettaz 1996]. Otherwise, the mobile node simulates the consumption of prey. It switches to the transitional phase *T5* in order to move back up to a height of 5 m–10 m [Arlettaz 1996]. When arriving at the target height the mobile node switches to *P5* and moves in circles with a radius of 2 m–4 m at the constant target height and a speed of 15 $\mathrm{km\,h^{-1}}$ for 10 s–20 s to imitate the prey consumption [Arlettaz 1996]. With a probability of 75 %, it switches back to *P2* to the search for prey, otherwise, it starts a period of rest.

With that, approximately 5 rests are accomplished in average, which corresponds to the findings in [Rudolph et al. 2009]. The mobile node transfers to transitional phase *T6*, where it moves to a height of 3 m–5 m. When arriving at the target height it switches to *P6* and stays stationary to resemble the bat resting underneath tree barks in proximity to the ground. The resting lasts for 1 min–130 min with a mean of 60 min. Summing up the average resting time a mobile node rests for 20 % of the total simulation time, which corresponds to the findings in [Rudolph et al. 2009]. Afterwards, the mobile node changes back to movement pattern *P2* to search for prey. The last and final movement pattern *P7* simulates the return to the roost. After approximately 5.5 h this phase is triggered, which has a behaviour similar to *P1* [Rudolph et al. 2009].

Throughout the whole simulation a mobile node accelerates and decelerates with $2\,\mathrm{m\,s^{-2}}$ and $4\,\mathrm{m\,s^{-2}}$, respectively, and the ascending and descending speed amounts to $1\,\mathrm{m\,s^{-1}}$ and $2\,\mathrm{m\,s^{-1}}$, respectively. Furthermore, a mobile node never moves in a complete straight line. With each position update, scheduled every 0.1 s, the direction of the movement changes by -90° to 90°, whereby this random change is normally distributed with a mean of 0° and a standard deviation of 30°.

In order to obtain reproducible results but still to exploit all the variations in the distribution of the needed random variables, we decided to run different mobility patterns using different seeds for the mobility-related random variables. Therefore, mobile nodes move distinctly in each repetition of the simulation, whereas their movements are the same for the n^{th} repetition of a certain configuration.

4.3. Validation of Model Implementation

In order to develop a realistic simulation model of the foraging behaviour of bats, we proceed in two steps. First, we assure a most realistic theoretical mobility model for the foraging behaviour of bats. Secondly, we validate the correctness of the implementation of this model. The first step to develop the theoretical model was to integrate all available information found in the literature. In a second step, biologists revised and extended the model with their knowledge and experience in order to develop the prior described mobility model. In order to ensure a correct implementation of the model we relied on statistical analysis of all mobility-related simulation results. In the following, the most important parameters are discussed and compared to the theoretical model.

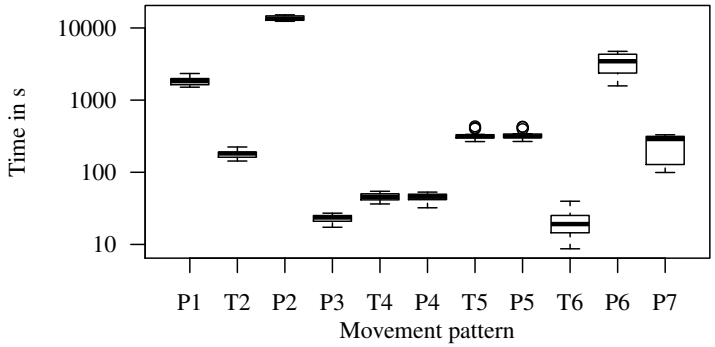


Fig. 5: The time a mobile node spends in the various movement patterns (cf. Table I).

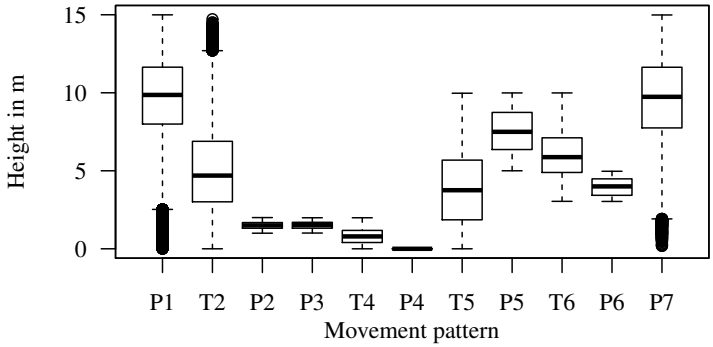
First, we analyze the partitioning of the simulation time into the different movement patterns shown in Figure 5. The results are plotted in the form of a boxplot. For each data set, a box is drawn from the first quartile to the third quartile, and the median is marked with a thick line. Whiskers extend from the edges of the box towards the minimum and maximum of the data set, but no further than one and a half times the interquartile range. Outliers are drawn separately.

With traditional radio telemetry, it is difficult to observe a bat throughout the whole activity period, hence, we do not have enough knowledge for a straightforward breakdown of a bat flight into the various movement patterns. Nevertheless, figures about the time a bat spends in some of the movement patterns can be derived and were integrated into the mobility model. According to Figure 5, a mobile node spends most of the time hunting in movement pattern *P2*, which is approximately 70 % of the total simulation time.

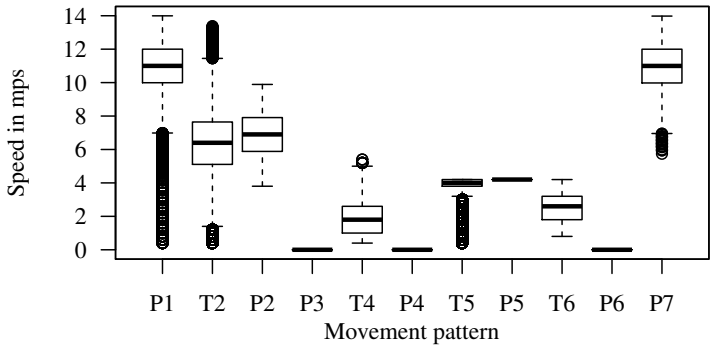
Moreover, Figure 5 shows that a bat rests in *P6* for about 1500 s–4800 s, which corresponds to 8 %–24 % of the total simulation time. Hence, the total resting time assumed by the theoretical model, i.e., 20 %, lies within this range. The time spent for prey consumption *P5* is on average 330 s per simulation run, which lies in a reasonable range given that a bat consumes 10–30 insects each within 10 s–20 s.

The distribution of the hunting grounds also influences the time a mobile node spends in between these locations, i.e., in movement pattern *P1*. The distance between the hunting grounds in our simulation is 1 km–5.4 km with an average of 2.8 km, which is in line with typical field research sites. Hence, with the given speed of 30 km h^{-1}–50 km h^{-1} mobile nodes need in average 200 s–340 s to move to a different hunting ground. Since mobile nodes visit up to eight hunting grounds, the average time spent between these areas, i.e., movement pattern *P1*, sums up to a maximum of 1600 s–2720 s, which corresponds to the values visualized in Figure 5.

Next, we analyze the parameters in which the various movement patterns differ the most, namely the flight height and speed. The two parameters are visualized for each movement pattern in Figures 6a and 6b, respectively. It can be seen that the height and speed while moving between hunting areas *(P1)* and returning to the roost *(P7)* are similarly distributed, which corresponds to the theoretical model, and lie mostly within the determined ranges. The few lower outliers are due to a starting height and speed outside the range. The height during the search for prey *(P2)* and the subsequent hovering in the air *(P3)* should obviously be identical with values in between 1 m–2 m. Moreover, during prey capture *(P4)* a mobile node is located on the ground, hence, the



(a) Height of the mobile nodes.



(b) Speed of the mobile nodes.

Fig. 6: Most influential parameters of the movement patterns.

height is fixed to a value of zero. Likewise, the height during prey consumption *(P5)* and resting *(P6)* has to be between 5 m–10 m and 3 m–5 m, respectively.

As Figure 6a confirms, these requirements are met in the simulation. Figure 6b shows a constant speed of zero during hovering *P3*, prey capture *P4*, and resting *P6*, which corresponds to the theoretical model. During prey consumption *P5* the speed is fixed to $15\,\mathrm{km\,h^{-1}}$ accordingly to the theoretical model. Finally, both Figures 6a and 6b show that the height and speed during the transitional phases *T2*, *T4*, *T5* and *T6* have values in between their preceding and succeeding movement patterns. The only exception is the speed during transitional phase *T4*, i.e., the change from hovering (*P3*) to prey capture (*P4*). These two phases have a target speed of $0\,\mathrm{km\,h^{-1}}$, whereas the transitional phase in between has a non-zero speed, which corresponds to Figure 6b.

We would like to point out that also the number of successful and failed catches corresponds to the model. Furthermore, the number of rests exactly matches the observations in the wild. Results from our simulation model are shown in Figure 7. As can be seen, the number of successful catches is about three times as much as the failed approaches.

4.4. Contact Opportunities

Using the presented mobility model, we can start investigating the contact opportunities between individual bats and a ground station. This property is crucial for the

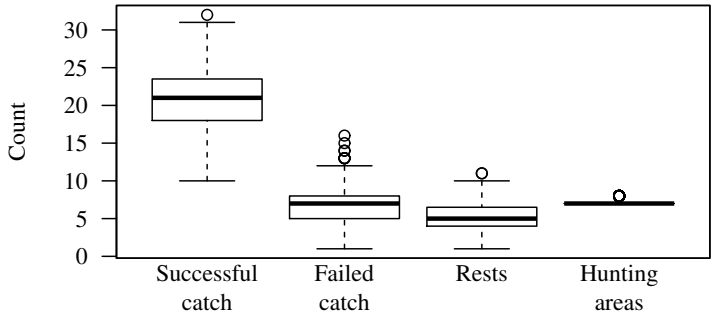


Fig. 7: Number of successful and failed captures, rests and visited hunting areas.

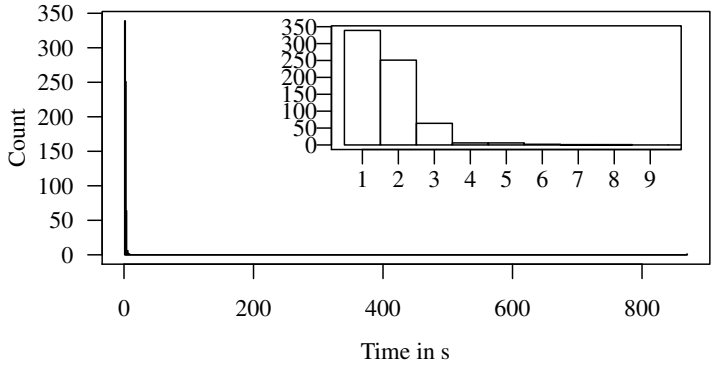


Fig. 8: Histogram of the contact times.

communication protocol from an overall perspective. We define contact opportunities as the time intervals in which communication between mobile nodes and the base network is possible. We investigated these contact times before finally assessing the performance of the EC based communication.

A histogram of the contact times is given in Figure 8. It summarizes the contact time observed per bat for 20 repetitions of the simulation using the described mobility model. In addition to the complete histogram the figure contains an excerpt of the contact times up to 10 s in order to provide a higher resolution for this range. The granularity of the contact time is set to 1 s, which is a simulation artifact since the transmission range of a mobile node towards the base network is checked once during every round of the duty cycle, which is set to be 1 s.

As the figure indicates, the contact time is mainly short-termed. Nearly 90 % of all contacts have a duration of less than 3 s. Since mobile nodes tend to leave the communication range of a base node quickly after entering, the amount of data that can be transmitted is limited.

A second measure of interest is the inter-contact time, i.e., the time intervals in which no communication between a mobile node and the base network has been possible. Figure 9 visualizes a complete histogram of the inter-contact times as well as a histogram enlarging the distribution for a range of 0 s–20 s, again summarizing all 20 repetitions. The distribution is heavy-tailed with an upper limit given by the total simulation time. The high peak at 1 s has already been discussed as being a simulation artifact. 80 %

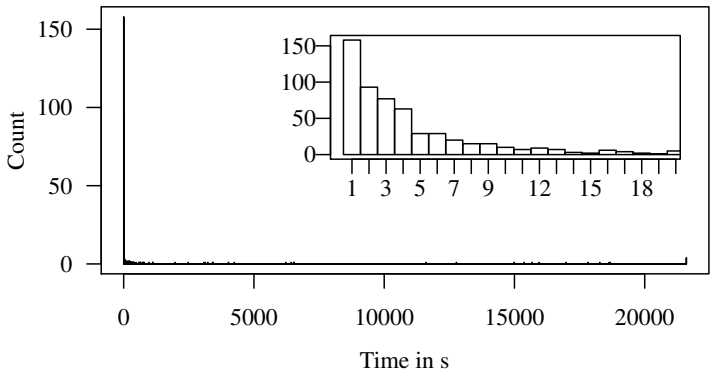


Fig. 9: Histogram of the inter-contact times.

of all inter-contact times have a duration of less than 20 s with an upward tendency towards 1 s.

5. COMMUNICATION PROTOCOL

In the following, we briefly explore the used communication protocols. Main emphasis is on the mobile-to-ground communication, for which we propose the use of Erasure Codes (ECs). Both protocols make heavy use of a multi-stage wake-up receiver ensuring that the micro controller and the digital radio transceiver are turned off most of the time. The introduction of the used hardware is out of scope for this paper but more information on the wake-up system can be found in [Dressler et al. 2015]. In contrast to our earlier work in [Mutschlechner et al. 2014b], we now use more realistic values related to the final hardware constraints.

5.1. Collecting Contact Information

Mobile nodes periodically broadcast beacons of 1 B containing their unique identifiers to inform nearby mobile nodes about their existence. The interval between two beacons was set to be 1 s. Upon reception of a beacon, the receiving mobile node stores this information to build up a rendezvous table.

Each row of this table consists of the 4 bit identifier contained in the beacon, the time when the first beacon from this node was received, and the time interval for which beacons were continuously received. Two beacons are considered to be received continuously if the difference of their arrival time is smaller than a pre-defined threshold, which is set to 5 s. The time stamp when the first beacon was received has a size of 21 bit, which reflects the envisioned deployment time of approximately 2 weeks, whereas 15 bits are reserved for the duration of this rendezvous since the maximum travel time of a bat is at most 6 h.

Each row of the rendezvous table is transmitted in an individual data chunk to the base network. The transmission of data chunks has to be reliable, whereas the successful reception of beacons is not as critical since the quantity of useful information within a beacon is comparably less than within a data chunk. Therefore, ECs are used only for data chunk transmissions.

5.2. Reliable Upload of Contact Data

When a mobile node is within the transmission range of at least one base node (this is determined by the wake-up signal powering up the digital radio transceiver and the micro controller), it tries to transmit the content of the rendezvous table. Obviously, no

data is transmitted if the rendezvous table is empty. Moreover, a data transmission is initialized only if the number or the age of table entries reaches a pre-defined threshold.

If no EC is used, the threshold for the number of entries is set to 1, otherwise the rendezvous table has to have at least 4 entries. This can be motivated by the fact that the encoding of a smaller data amount restricts the variation of redundancy that can be added, e.g. for 1 entry either no redundancy or 100 % redundancy can be added. However, if at least one rendezvous table entry is older than 60 s the data is transmitted independent of the rendezvous table size to guarantee the transmission of all data, i.e., to omit starvation. If ECs are used and there are no encoded chunks already prepared the encoding process is triggered before the data transmission.

Due to the severe energy restrictions a duty-cycling mechanism is applied where only a short time window is reserved for the data transmissions. This window size forms an upper limit for the packet size, which is approximately 8 B. Since such a window is reserved every 1 s, we have an actual throughput of $8\,\mathrm{B\,s^{-1}}$. The first 5 B of a data packet are reserved for the payload, which is the size of one rendezvous table entry.

In case of using ECs, the payload may contain also one encoded chunk. The remaining 3 B are used for the mobile node's identifier of 4 bit, 1 B meta data of the EC and a checksum. In case of using on-demand retransmissions, a base node receiving such a packet immediately transmits an acknowledgment to inform the mobile node about the successful reception. In case the mobile node does not receive this information until the next transmission window it resends the same packet.

Due to the mobile nodes' high speed and the rapidly changing environment the contact times between mobile nodes and base nodes can be very small (cf. Section 4.4). Therefore, no carrier sensing techniques are performed prior to transmission since this could prevent the mobile nodes from sending data before exiting the transmission range. Moreover, the mobile nodes are highly constrained in terms of computational power and energy, hence, very complex protocols are not applicable. Instead, the wake-up receiver powered up by a signal from the base nodes initializes the data transmission.

Upon receiving a data chunk from a mobile node, base nodes store this information for decoding. For each received data chunk, the central station records its reception and, if the threshold for a successful decoding is reached, it tries to recover the original data.

The communication channel resulting from this specific mobility model represents the channel in the real deployment scenario we are facing. To simulate realistic data transmissions over the wireless channel we chose a free-space path-loss propagation model as basis for the communication channel:

$$L = \frac{1}{d^3}\left(\frac{c}{4\pi f}\right)^2 , \tag{2}$$

The signal gets attenuated with the distance d between sender and receiver with respect to the carrier frequency f and the constant speed of light c. The model assumes no obstacles, i.e., no reflecting ground, between the sender and the receiver. We have chosen such a model instead of, for example, the two-way interference model since the distance of the nodes is negligible low with respect to their height and most signal disturbing factor are obstacles such as trees. In order to simulate these interferences, we integrated a log-normal shadowing with mean 0.5 and a standard deviation of 0.25 on top of it to simulate objects obstructing the signal. This can be motivated by the fact that we are facing multiple shadowing objects in the envisioned deployment scenario, i.e., trees. Each such object contributes a random multiplicative factor to the shadowing, which leads to a normal distribution when converted to dB [Andersen et al. 1995].

It is well known that simulation models may lead to imprecise (or even wrong) conclusions when abstracting too many details from reality. We therefore conducted a series of experiments in the wild to explore the radio propagation characteristics to be

expected. These experiments were focusing on Wi-Fi and IEEE 802.15.4 sensor nodes in different types of forest environments [Mutschlechner et al. 2013]. We used the results to calibrate our simulation models.

The employed frequency in our simulations is 686 MHz, which is the target frequency for the data communication of the mobile nodes as described in [Dressler et al. 2015] – this paper also details the technological background on the wake-up system. The transmission power of the mobile nodes is set to 10 mW. The wakeup range, i.e., the maximum distance to a base node that triggers the wake-up receiver, is approximately 10 m. The communication between the various nodes is not synchronized in any way. Coordinating the communication would require an additional overhead, which is not desired in our highly energy-constrained scenario if the issue is not indispensable. Since we have a sparse distribution of senders, i.e., a small amount of nodes compared to the size of the area, and a very limited transmission range due to the energy constraints, we have hardly no overlapping transmission. This is also shown in our evaluation in Section 6.2.1. Hence, we have decided not to adapt any synchronization techniques.

6. PERFORMANCE EVALUATION

The performance evaluation is twofold. First, we analyze the system performance of the EC implementation on the real system focusing on energy consumption and computational overhead. Secondly, we investigate the usability of ECs in the communication system using a simulation setup based on three metrics: reliability, energy consumption, and delay.

6.1. System Performance

Erasure codes are commonly used in systems with adequate performance characteristics (i.e., high processing power and sufficient available memory). Due to hardware restrictions (e.g., scarce memory) as a result of limited size on mobile nodes in the BATS scenario, one has to cope with certain requirements for all software components that run on top. First of all, it was necessary to adapt both selected RS implementations of the encoding functions (cf. Section 3.1) to run on our embedded target evaluation platform. In this section, we focus on encoding functions, because the (more complex) decoding process happens on the more powerful ground nodes or even at a central node. Hence, we can use the original decoding routines with marginal changes.

In the following, we first introduce the target evaluation platform, which we refer to as the *device under test* (DUT). Subsequently, we describe our measuring platform and its underlying methodology and, finally, we present and discuss the measuring results.

6.1.1. Evaluation Setup. To find the best-fitting evaluation platform, it was necessary to analyze all our requirements regarding hardware capabilities, system software, and application code. From a hardware perspective, erasure-code algorithms make use of rather complex operations. However, despite this set of features, the platform needs to be very energy efficient. This combination can be only achieved if power-saving modes and other energy-saving features are provided. Therefore, as a prototype we used a hardware platform with comparable technical specifications to the envisioned target platform. All energy measurements were performed on the Freescale FRDM-KL02Z development platform [Freescale Semiconductor, Inc. 2013] with 32 kB flash memory and 4 kB SRAM. Furthermore, it provides different ultra-low–power modes and ARM's Cortex-M0+ 32-bit processor with the, at the time of writing, smallest and most energy-efficient package design available.[2] The processor operates at up to 48 MHz and has built-in features such as a two-stage pipeline and a memory protection unit. Figure 10

[2]http://www.arm.com/products/processors/cortex-m/cortex-m0plus.php

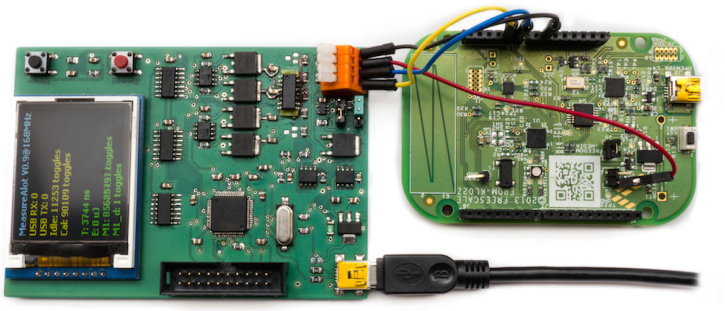


Fig. 10: The evaluation platform under test (Freescale FRDM-KL02Z) on the right, which is connected with the energy measuring device on the left.

shows the complete evaluation setup, consisting of the DUT, our custom measuring device, whose functioning is described in the following subsection, and the complete wiring between the DUT and the measuring device.

Our scenario poses strict constraints which besides limited hardware capabilities requires tailored software components. At software level, fulfilling these requirements largely concerns the operating system; that is, our approach requires an operating system that is aligned to both the requirements of our application software and the target hardware platform. Therefore, from available real-time operating systems, we chose SLOTH [Hofer et al. 2009], a light-weight real-time operating system. It fits particularly well because of its resource-efficient operating mode by means of utilization of hardware features lying underneath. In particular, by letting hardware interrupt subsystems execute most of the scheduling and dispatching activities, not only a great speedup but also high savings in memory usage are achieved. Hence, we implemented an application on top of the SLOTH kernel that contains EC encoding functions, which involved only a few, but necessary adaptations specific to our target platform. For example, the limited memory available on our target platform led to special adaptations to the Vandermonde-matrix–based implementation. More precisely, the encoding process maps data packets to a series of redundancy packets, which entails a multiplication on a finite field with 256 elements. At this point the original implementation uses pre-calculated values, which do not fit into memory, and our implementation therefore relies on online calculations, which have a small memory footprint: The multiplication values are calculated on demand with an implementation of the Russian peasant multiplication algorithm [Bunt et al. 1988, p. 13]. All SLOTH applications to be measured were compiled with the cross compiler from GNU Tools for ARM Embedded Processors 4.7 (Q2 2013) and SLOTH's default optimization level that aims to optimize for code size (`-Os`).

6.1.2. Measurement Methodology. Retrieving precise energy values for low-power devices as required in the BATS project is very tedious, or even impossible, with usual energy-measurement approaches. Particularly, current best-practice approaches require to use hardware (e.g., micro controllers with analog-to-digital converters) with the drawback of having sampling rates that are too low to provide fine-grained accurate measurement results. Alternatively, one could use oscilloscopes combined with differential probes, however, they are limited in multiple ways (e.g., not intended for measurements over longer periods of time) and generally cumbersome to handle. Therefore, we used a self-developed measuring device (cf. Figure 10) for all run-time and energy measurements, which we have been developing in the course of earlier [Hönig et al. 2011] and

ongoing [Hönig et al. 2014] work. This device is superior to conventional approaches that measure, for example, the voltage drop across a resistor (shunt).

The main components of our measurement device are composed of a current mirror, which is implemented using PNP transistors, and an RS flip-flop that is responsible for *current-to-frequency* conversion. The current mirror duplicates the DUT's current, and two additional capacitors are being charged and discharged on a rotating basis. The RS flip-flop's output Q signals whether the first or second capacitor is currently being charged (while the other one is discharged at the same time). As a consequence, the flip-flop's output continuously toggles from a logical 0 to a logical 1 and vice versa, depending on the capacitors' voltage levels. Since these 0-to-1 and 1-to-0 transitions are directly proportional to the current drawn, deriving and calculating the DUT's energy consumption becomes possible.

This way, measuring a program's energy consumption is simplified as follows: At first, the measuring device needs to be connected with the DUT. In case of the FRDM-KL02Z, it is viable to isolate the MCU's (i.e., both processor and memory) power drain from the rest of the system, which minimizes noise generated by other system components that would potentially distort measurement results. A second and third connection link between the DUT and the measuring device is used for signaling the start and end of a measurement as well as to wait until the measuring device is ready for processing new measurements, respectively (the fourth wire is connected to ground). The program code under test (i.e., our SLOTH application) just has to be prepared for toggling (at measurement start and end) a specific GPIO pin that is connected with the measuring device. When a measurement run is complete, the resulting values can automatically be retrieved by the workstation that is also connected with the measuring device.

6.1.3. Measurement Results. The evaluation is decomposed into three parts: First, we present runtime information to get estimates for average computing times that are required in the MCU. Secondly, since the encoding algorithms are used in environments with not only limited battery lifetimes but also very limited available memory, we measured the memory usage of the implemented erasure-code algorithms. Thirdly, we present concrete energy values for executing a single encoding function. Measurements for all these metrics (runtime, memory, energy) were performed for both ported RS-code implementations, that is, the Cauchy- and the Vandermonde-matrix–based version.

To measure runtime and energy consumption, for each configuration (i.e., specific code rate) multiple measurements have been conducted. Within one measurement run, each encoding function is being executed multiple times; that is, the resulting value for one encoding function is computed by simply dividing the measured value by the number of actual executions. The values shown are average values over all measurement runs. As standard deviations and average errors are very small, they are not plotted in the diagrams. The number of original data packets is constant (i.e., 4), whereas the number of redundancy packets is varied between 1 ($r = \frac{4}{5}$) and 8 ($r = \frac{4}{12}$).

Timing Analysis. As our measuring device also provides high accuracy in terms of timer resolution, we used it for measuring the encoding functions' runtimes. Its high precision is achieved by means of a 32-bit timer with an internal clock of 168 MHz, which results in a resolution of not more than 5.95 ns. A comparison of the two erasure-code implementations with respect to execution times is shown in Figure 11. As can be seen, the run times of both algorithms increase linearly with the number of redundancy packets, which is reasonable since each encoding function contains a single loop that depends on this number ($\rightarrow$ complexity: $\mathcal{O}(n)$). The percentages on top of the bars representing the Vandermonde-matrix–based implementation reflect the runtime overhead compared to the other type of RS-code implementation. With these values, it is more obvious to see that as the code rate increases, the runtime of the Cauchy-matrix–based

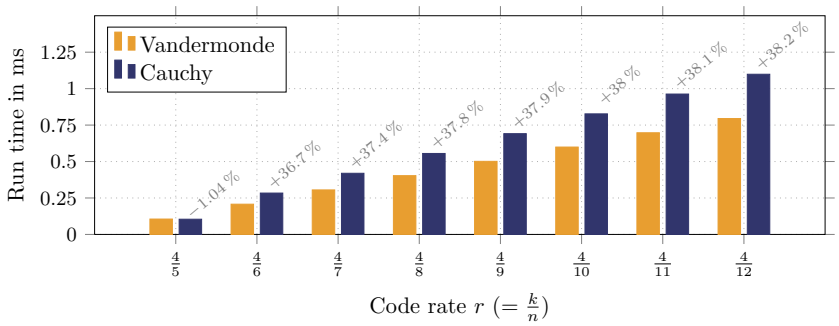


Fig. 11: Run-time comparison between the Cauchy- and the Vandermonde-matrix–based RS-code implementation. The values represent a single execution of the corresponding encoding function.

Implementation Type	**Code Rate** ($= \frac{k}{n}$)	**Memory Allocation**			
		Data	Stack [byte]	Total	Relative [%]
Vandermonde	$\frac{4}{5}$	20	21	41	1.00
	$\frac{4}{6}$	24	21	45	1.10
	$\frac{4}{7}$	28	21	49	1.20
	$\frac{4}{8}$	32	21	53	1.29
	$\frac{4}{9}$	36	21	57	1.39
	$\frac{4}{10}$	40	21	61	1.49
	$\frac{4}{11}$	44	21	65	1.59
	$\frac{4}{12}$	48	21	69	1.68
Cauchy	$\frac{4}{5}, \frac{4}{6}, \ldots, \frac{4}{12}$	318	56	374	9.13

Table II: Memory-consumption values for encoding functions of both Cauchy- and Vandermonde-matrix–based RS-code implementations. “Data” indicates memory allocated for data residing in the data segment (i.e., global and static local variables), whereas “Stack” summarizes allocated memory for all local variables. The relative value reflects the respective implementation’s total memory consumption in relation to the total available memory (= 4 kB) for storing variables at run-time on the FRDM-KL02Z.

implementation increases slightly faster than the second implementation; that is, in general, its execution time is higher (at least 36.7 %; with one exception at code rate $\frac{4}{5}$, where both algorithms perform almost identically) and grows faster with more redundancy packets. However, with maximums of 1.1 ms (Cauchy-matrix–based version) and 0.8 ms (Vandermonde-matrix–based version) at code rate $r = \frac{4}{12}$, both algorithms are suitable when considering the time window of 1 s which is reserved for sending a rendezvous-table entry (cf. Section 5.2).

Memory-Consumption Analysis. Keeping the required memory consumption at a minimum is important in our scenario. Table II shows memory-consumption values for

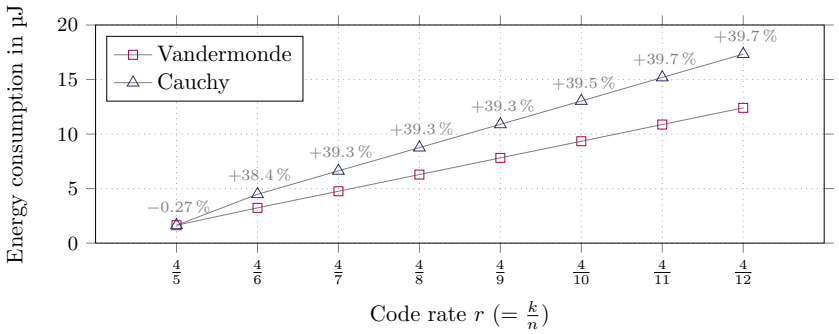


Fig. 12: Energy-consumption comparison between Cauchy- and Vandermonde-matrix–based RS-code implementations.

both RS-code implementations and all previously used code rates (encoding functions only, that is, without any operating-system overhead[3]). For both of them, the maximum allocated stack size per configuration remains constant. However, in contrast to the Cauchy-matrix–based implementation, the memory allocated in the data segment for the Vandermonde-matrix–based implementation (41 B–69 B) increases with the number of redundancy packets. This is a result of different encoding matrices that are required for each configuration of both data and redundancy packets. In general, the number of columns of theses matrices equals the number of original data packets to be encoded, whereas the number of rows is the sum of redundancy- and original data-packet counts. Nevertheless, for all analyzed code rates the Vandermonde-matrix–based implementation consumes only between approximately one ninth ($\frac{41\,\mathrm{B}}{374\,\mathrm{B}}$) and one fifth ($\frac{69\,\mathrm{B}}{374\,\mathrm{B}}$) compared to the Cauchy-matrix–based implementation.

Energy-Consumption Analysis. Figure 12 illustrates the energy behavior of both RS-code implementations, again for the same set of code rates used for runtime and memory analyses. Each value corresponds to the energy value for one execution of the respective encoding function. Similar to the runtime behavior, the energy consumption increases linearly with the number of redundancy packets. Furthermore, the Cauchy-matrix–based implementation consumes for code rates with at least two redundancy packets about 40 % more energy than the Vandermonde-matrix–based implementation (1.6 µJ–17.3 µJ versus 1.6 µJ–12.4 µJ). Since for many instruction types[4], a strong correlation between runtime and energy behavior can be observed, this is in line with our expectations. Hence, with the exception of code rate $\frac{4}{5}$ – were both algorithms behave almost identically in terms of runtime and energy –, the Vandermonde-matrix–based implementation also performs better with regard to energy consumption.

All metrics (especially the energy metric) indicate that the Vandermonde-matrix–based implementation outperforms the Cauchy-matrix–based implementation. We conclude that from the system perspective, this encoding scheme is to be recommended for the BATS scenario.

[3]The operating-system overhead is a constant value of 1584 B for both implementations, as the number of tasks and enabled features is the same in either case.

[4]It is possible to estimate a program's energy consumption by examining its executed instruction counts and joining these values with previously established instruction-specific energy values (e.g., as done in the SEEP energy-estimation framework [Hönig et al. 2011]).

6.2. Communication Performance

In the following, we investigate the performance of the communication protocols in more detail. We concentrate on the reliability vs. overhead issue but also look into energy consumption and delay questions. All messages will be transmitted without acknowledgements due to time (contact) and energy (additional downlink) constraints. In the simulations, however, we also explored ARQ as a baseline, indicating the potential reliability improvements.

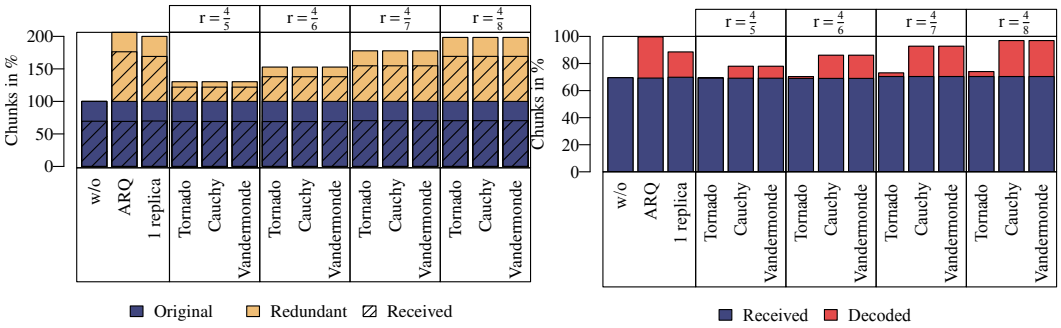
For all experiments, we used the already explained simulation setup. For simulating the wireless channel, we used the MiXiM framework [Köpke et al. 2008], which provides all the means for accurate wireless simulation and the integration of mobility models. The movements of mobile nodes are based on a predefined mobility model that resembles the flying behavior of bats, whereas base nodes are stationary with a fixed position across all simulations. The mobility model of the mobile nodes has been described and discussed in Section 4.2.

In order to get a better understanding of the communication performance, we assessed the protocol using different packet error rates. In addition to the described channel model, we configured different packet error rates of 10 %, 20 %, and 40 %. This helps to identify the protocol behavior even in worst case communication scenarios in the forest.

For each configuration 20 repetitions were performed. Each repetition of one configuration is initialized with a unique random seed, however, the n^{th} repetition of each configuration has the same n^{th} random seed. The variance of most recorded metrics (except for the delay measures) was so small that we only plot the average values in the respective figures.

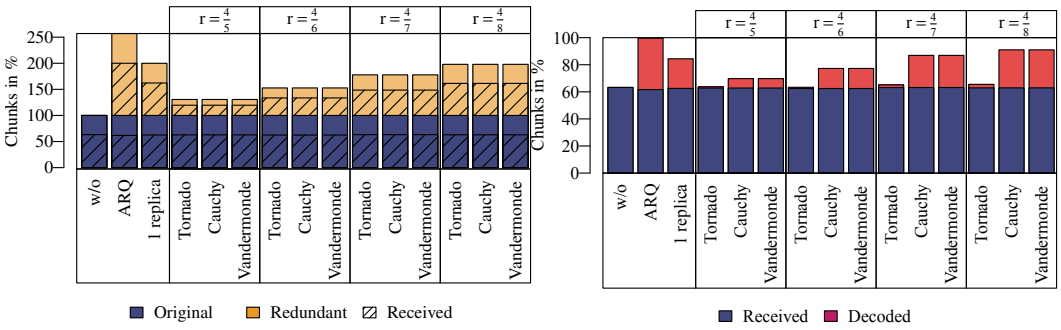
6.2.1. Transmission Overhead and Achieved Reliability. To investigate the quality of the channel, we consider the relationship between sent and received chunks. All chunks received erroneously or not received at all were considered to be lost. Figures 13a, 14a and 15a show this relationship by visualizing both the amount of sent chunks containing original data as well as the amount of sent chunks containing redundant information for our selected packet error rates of 10 %, 20 %, and 40 %, respectively. Furthermore, the amount of received chunks of both types is indicated by shaded areas. All numbers are relative to the amount of sent data, i.e., 100 % corresponds to the amount of sent data. The improvement of ECs on the transmission reliability is analyzed based on the ratio of received and recovered data with respect to the amount of sent data. This relationship is visualized in Figures 13b, 14b and 15b, again for the three packet error ratesi of 10 %, 20 %, and 40 %, respectively. The amount of received (decoded) data is visualized as a blue (red) bar. Please note that the amount of recovered data is the sum of the received and decoded chunks, i.e., the total height of the single columns. All numbers are normalized to the amount of sent data. The variance for all measures was negligibly small.

Apparently, the amount of sent original chunks stays constant, i.e., it is independent of both the reliability improvement technique and the packet error rate. However, the amount of sent redundant chunks varies with these two factors. Clearly, when using no reliability improving technique there are no redundant chunks sent at all. The amount of sent redundant data when using ECs depends solely on the code rate, i.e., the packet error rate does not have any impact since the amount of redundant data is not influenced by the reception rate. According to the code rate of $r = \frac{4}{5}$, $r = \frac{4}{6}$, $r = \frac{4}{7}$ and $r = \frac{4}{8}$, the overhead spent in sending redundant data amounts to 25 %, 50 %, 75 % and 100 %, respectively. By using the replicated sending approach we obtain results similar to the ECs with the highest code rate, i.e., their effort spent in sending redundant data. The approach of using ARQ, however, depends heavily on the packet error rate. This can be explained with its dynamic sending of redundant chunks, which increases with



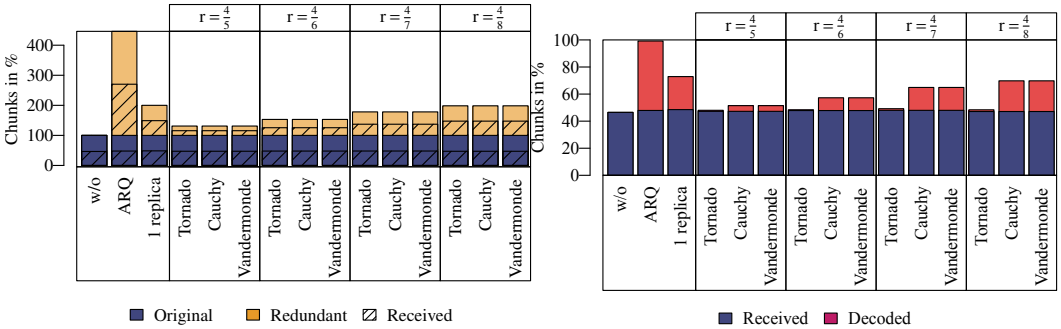
(a) Chunks sent vs. received (b) Chunks received vs. decoded

Fig. 13: Transmission success rate for a packet error rate of 10 %



(a) Chunks sent vs. received (b) Chunks received vs. decoded

Fig. 14: Transmission success rate for a packet error rate of 20 %



(a) Chunks sent vs. received (b) Chunks received vs. decoded

Fig. 15: Transmission success rate for a packet error rate of 40 %

a decreasing reception rate. With a packet error rate of 10 % an overhead of 110 % is spent in sending redundant data. If the packet error rate is increased up to 20 % and 40 %, the overhead is raised up to 150 % and 300 %, respectively.

For both original and redundant data types the reception rate changes only with the varying packet error rate, i.e., it stays constant regarding the same packet error rate. With a packet error rate of 10 % the ground network is able to receive about 70 % of the overall amount of sent data. With a rising packet error rate of 20 % and 40 % the reception rate drops to 62 % and 47 %, respectively.

Considering this constant reception rate, we argue that the number of packet transmissions does not influence the reception of packets, i.e., transmissions do not notably interfere with each other. Hence, we conclude that the channel is not saturated at all, which is due to the limited transmission range of mobile nodes and their sparse distribution. The high packet loss is mainly caused by the highly unreliable channel. These characteristics correspond to the channel we are facing in the real deployment scenario where the fast moving bats and the very heterogeneous and rapidly changing environment lead to a highly varying channel quality.

Apparently, without using ECs or replication techniques we are not able to recover any data not received by the base network, independent of the packet error rate. A notable result is the performance of the ARQ approach. Here we observe a constant recovery rate of almost 100 %, however, the suitability of this technique is raised to question when taking into consideration the amount of sent data as discussed before.

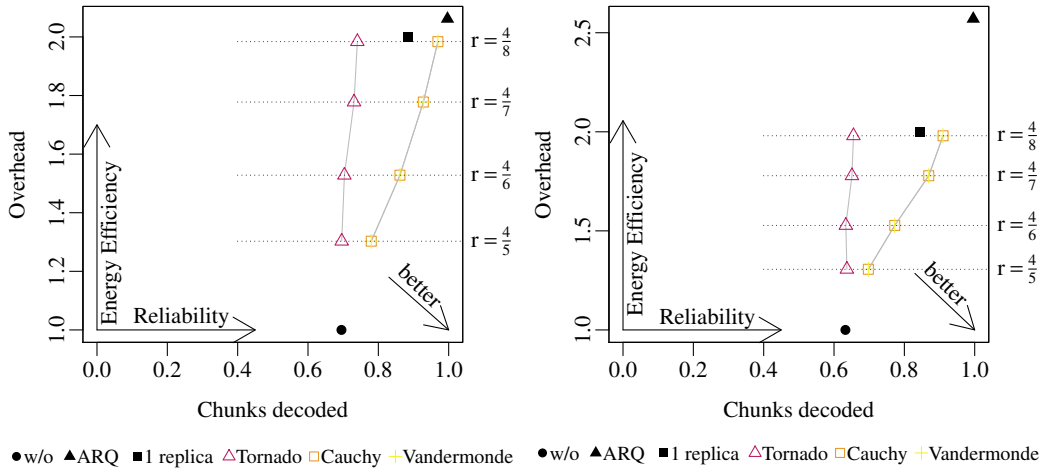
Using simple duplicates (full replication approach) the amount of recovered data increases with a decreasing reception rate. With a packet error rate of 10 % the base network is able to decode 18 % of the data from the redundant chunks, so in total 88 % of the data can be recovered. With an increased packet error rate of 20 % and 40 % the ground network is able to decode 22 % and 24 %, respectively, which corresponds to a total recovery rate of 84 % and 72 %, respectively. However, the drawback is a highly increased power consumption.

When using ECs, we can observe a steadily growing amount of recovered data when increasing the overhead in terms of code and therefore data rate, Furthermore, we see a huge difference between the performance of Tornado and the two RS codes Cauchy and Vandermonde.

Tornado slightly increases the amount of recovered data, which means redundant data does not noticeably improve the reliability. Even with a code rate of $r = \frac{4}{8}$, Tornado performs worse than replicated sending, although the same amount of chunks are transmitted. The poor behavior is mainly due to the application scenario using the code on application layer as well as using rather short messages. In our scenario, the different encoding algorithm of Tornado cannot provide the same performance as RS codes. Furthermore, Tornado is designed to work on a binary erasure channel, which means losses are assumed to be equally distributed. However, the simulations show that base nodes are facing bursty losses.

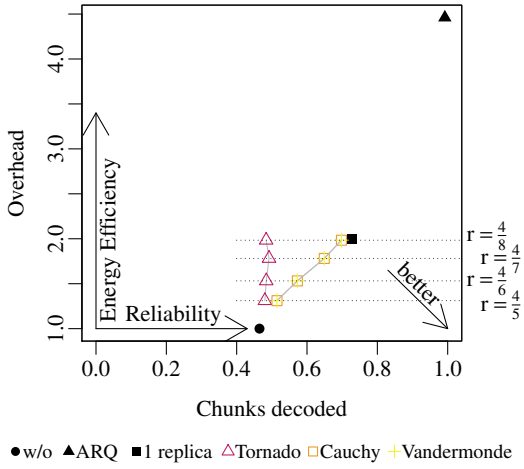
Cauchy and Vandermonde perform exactly the same regarding reliability, which is due to the similar encoding and decoding algorithms. At a code rate of $r = \frac{4}{8}$ both can decode 27 %, 28 % and 23 % of the sent data with respect to the packet error rates of 10 %, 20 % and 40 %. This shows that below a certain reception rate ECs perform poorly since not enough data is received to decode successfully. Compared to the replicated sending approach ECs perform better above a reception rate of 50 %. If this threshold is undershot the performance is reversed.

However, assuming a packet error rate of only 10 % and 20 %, Cauchy and Vandermonde outperform the replicated sending approach even with a code rate of $r = \frac{4}{7}$. They are able to recover 24 % and 22 %, respectively, although the overhead spent in sending



(a) Packet error rate of 10 %

(b) Packet error rate of 20 %



(c) Packet error rate of 40 %

Fig. 16: Reliability of the different transmission strategies versus their energy efficiency in terms of the necessary number of packet transmissions.

redundant data is decreased by 25 %. This shows that RS codes clearly outperform replicated sending in terms of reliability provided that the reception rate is above 50 %.

With an even smaller data rate of $r = \frac{4}{6}$ and $r = \frac{4}{5}$ the RS codes are able to decode 17 % and 9 %, respectively (packet error rate of 10 %). An increased packet error rate results again in a poorer performance.

6.2.2. Energy Efficiency. The usage of ECs and replicated sending inevitably increases energy consumption. Primarily the sending of redundant chunks drains energy, however, in the former case the execution of the encoding algorithm has to be taken into consideration as well. We have already discussed the coding efficiency of different ECs and the resulting system performance in Sections 3 and 6.1, respectively.

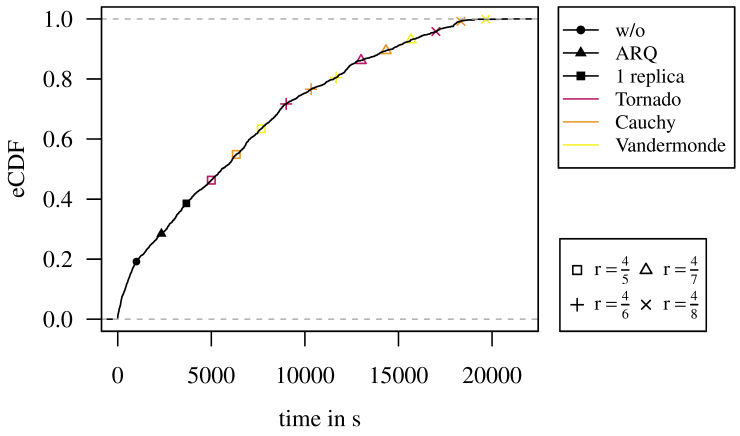


Fig. 17: eCDF of the data delay up to the total simulation time of 21 600 s; plotted are the results for the 10 % packet error rate. The plot shows macroscopic effects only for received messages, i.e., packet loss is not shown in the graph.

As the main energy-draining task is the sending of redundant chunks, we focus here on the comparison between the resulting overhead vs. the reliability improvement as a main metric (more detailed information on the energy consumption of selected ECs has been provided in Section 6.1). This trade-off between the improved reliability and the overhead caused by redundant chunks is outlined in Figure 16. The graphs summarize the results presented in the previous subsection and give an overview of the suitability of the presented reliability improvement techniques. As we move from left to right in the graphs, reliability measured against the amount of recovered data increases, whereas moving from bottom to top the energy efficiency decreases with an increasing overhead. The theoretical optimum would be on the indicated bottom-right corner, meaning that no additional energy is spent by means of overhead in the packet transmission but still all chunks can be received successfully. For reference, also the non-replicated sending is indicated, obviously not inducing any overhead but at the cost of very low reliability.

When considering the ARQ version, we reach 100 % success rate but the overhead grows to more than 450 %. As was already observed, Tornado as well as replicated sending perform poorly compared to the two RS codes. Here, the ratio between overhead and reliability can be configured by means of the code rate. Substantial improvements can be achieved already at a code rate of $r = \frac{4}{5}$ and reach very high reliability for $r = \frac{4}{8}$.

So, the remaining question as to which data rate for Cauchy and Vandermonde is most feasible strongly depends on the final application. Replication is also achieving good (even though slightly worse) results in terms of communication reliability compared to ECs but the energy overhead is too high for our system. In general, it holds that if a high level of reliability must be achieved, this comes to the cost of a reduced network lifetime [Dietrich and Dressler 2009]. According to our results, ECs provide the best compromise.

6.2.3. Communication Delay. The final measure to compare the various performance improvement techniques is the duration from the creation of a rendezvous table entry until a base node receives or recovers the data, i.e., the overall communication delay. Only the delay of data received by at least one base node is taken into consideration. We assumed that, given the rather short contact times and the substantial overhead of ECs for higher code rates, the resulting delay will clearly increase.

An Empirical Cumulative Density Function (eCDF) of the distribution of the data delay up to the total simulation time of 21 600 s is plotted in Figure 17. Only macroscopic effects, i.e., microscopic effects caused by ARQ, replicas, coding are not visible.

The figure shows, on the one hand, a highly varying data delay between a few milliseconds up to the total simulation time; the median is at about 6600 s. This can be explained by considering the contact and inter-contact time. Mobile nodes have a short inter-contact time, hence, even if they are not in transmission range of the base network they reenter after short time periods and are able to transmit data with small delays. However, the high upwards outliers increase the data delay by orders of magnitude.

On the other hand, the figure disproves our prior assumption of a negative impact of ECs on the data delay. They show clearly that the data delay is independent of the used algorithm and the packet error rate, i.e., the data delay is independent of the reliability improvement technique, the data rate and the reliability of the channel. From that we conclude that the negative impact of the overhead introduced by ECs is negligibly low compared to the influence of the specific contact and inter-contact time.

We only plotted the measured delays for a packet error rate of 10 %. The observed values for packet error rates of 20 % and 40 % show the same results.

In conclusion, it can be said that the that delays on the wireless link as well as delays for retransmissions are negligibly small compared to the macroscopic effects. Thus, we can fully concentrate on the reliability and the overhead and simply ignore the resulting additional delays of the algorithms.

7. CONCLUSION

We studied the potentials of using Erasure Codes (ECs) for communication in a very challenging sensor network scenario: monitoring bats in the wild. For this, we first developed a new mobility model to accurately predict the contact times and durations of an individual bat with a ground station in their natural habitat. This model has been validated using observations described in the respective literature. It turns out that this scenario features rather short contact times between the mobile nodes and stationary base nodes that are used to upload contact information a bat collected for further processing. Given the varying radio channel due to the speed of the animals and the challenging forest environment, the use of reliable communication techniques is to be recommended. Forward error correction using ECs turned out to be a promising solution.

We identified three feasible ECs and studied their performance in detail. As a baseline, we simply send the collected data with no additional reliability improvement techniques in place as well as full replication, i.e., sending all data items twice, and on-demand retransmissions. We were able to show that the nearly-optimal EC Tornado does not show a noticeable improvement on the data transmission, whereas the selected Reed-Solomon (RS) codes Cauchy and Vandermonde substantially increase the reception rate. This improvement comes with the cost of an overhead due to additional data messages that need to be sent; yet, the RS-based ECs clearly outperform simple data replication. Based on measurements on a real hardware platform, we were able to show that the computational overhead for encoding is feasible – the same can be noted for the energy consumption on the nodes. In summary, the results show that ECs provide a significant reliability improvement with an acceptable overhead for our extremely energy-constrained hardware platform.

References

Jorgen Bach Andersen, Thedore S. Rappaport, and Susumu Yoshida. 1995. Propagation measurements and models for wireless communications channels. *IEEE Communications Magazine* 33, 1 (Jan. 1995), 42–49. DOI: http://dx.doi.org/10.1109/35.339880

Georgios Angelopoulos, Arun Paidimarri, Anantha P Chandrakasan, and Muriel M'edard. 2013. Experimental Study of the Interplay of Channel and Network Coding in Low Power Sensor Applications. In *IEEE International Conference on Communications (ICC 2013)*. IEEE, Budapest, Hungary, 5126–5130. DOI:http://dx.doi.org/10.1109/ICC.2013.6655396

David Anthony, William P. Bennett, Mehmet C. Vuran, Matthew B. Dwyer, Sebastian Elbaum, Anne Lacy, Mike Engels, and Walter Wehtje. 2012. Sensing Through the Continent: Towards Monitoring Migratory Birds Using Cellular Sensor Networks. In *11th ACM/IEEE International Symposium on Information Processing in Sensor Networks (IPSN 2012)*. ACM, Beijing, China, 329–340. DOI:http://dx.doi.org/10.1145/2185677.2185747

Raphael Arlettaz. 1996. Feeding behaviour and foraging strategy of free-living mouse-eared bats, Myotis myotis and Myotis blythii. *Animal Behaviour* 51, 1 (1996), 1–11. DOI:http://dx.doi.org/10.1006/anbe.1996.0001

Doris Audet. 1990. Foraging Behavior and Habitat Use by a Gleaning Bat, Myotis myotis (Chiroptera: Vespertilionidae). *Journal of Mammalogy* 71, 3 (1990), 420–427. http://www.jstor.org/stable/1381955

Christian R. Berger, Shengli Zhou, Yonggang Wen, Peter Willett, and Krishna Pattipati. 2008. Optimizing Joint Erasure- and Error-Correction Coding for Wireless Packet Transmissions. *IEEE Transactions on Wireless Communications* 7, 11 (Nov. 2008), 4586–4595. DOI:http://dx.doi.org/10.1109/T-WC.2008.070581

Johannes Blömer, Malik Kalfane, Richard Karp, Marek Karpinski, Michael Luby, and David Zuckerman. 1995. *An XOR-Based Erasure-Resilient Coding Scheme*. Technical Report TR-95.048. International Computer Science Institute, Berkeley.

Lucas N. H. Bunt, Phillip S. Jones, and Jack D. Bedient. 1988. *The Historical Roots of Elementary Mathematics*. Dover Publications. 336 pages.

Isabel Dietrich and Falko Dressler. 2009. On the Lifetime of Wireless Sensor Networks. *ACM Transactions on Sensor Networks (TOSN)* 5, 1 (Feb. 2009), 1–39. DOI:http://dx.doi.org/10.1145/1464420.1464425

Falko Dressler, Bastian Bloessl, Martin Hierold, Chia-Yu Hsieh, Thorsten Nowak, Robert Weigel, and Alexander Koelpin. 2015. Protocol Design for Ultra-Low Power Wake-Up Systems for Tracking Bats in the Wild. In *IEEE International Conference on Communications (ICC 2015)*. IEEE, London, UK, 6345–6350. DOI:http://dx.doi.org/10.1109/ICC.2015.7249335

Falko Dressler, Simon Ripperger, Martin Hierold, Thorsten Nowak, Christopher Eibel, Björn Cassens, Frieder Mayer, Klaus Meyer-Wegener, and Alexander Koelpin. 2016. From Radio Telemetry to Ultra-Low Power Sensor Networks - Tracking Bats in the Wild. *IEEE Communications Magazine* (2016). to appear.

Freescale Semiconductor, Inc. 2013. *FRDM-KL02Z User Manual, Revision 0*. Freescale Semiconductor, Inc.

R.G. Gallager. 1962. Low-density parity-check codes. *IRE Transactions on Information Theory* 8, 1 (Jan. 1962), 21–28. DOI:http://dx.doi.org/10.1109/TIT.1962.1057683

Antonio-Javier Garcia-Sanchez, Felipe Garcia-Sanchez, Fernando Losilla, Pawel Kulakowski, Joan Garcia-Haro, Alejandro Rodriguez, Jose-Vicente Lopez-Bao, and Francisco Palomares. 2010. Wireless Sensor Network Deployment for Monitoring Wildlife Passages. *Sensors* 10, 8 (2010), 7236–7262. DOI:http://dx.doi.org/10.3390/s100807236

Anders Hedenström, L. Christoffer Johansson, and Geoffrey R. Spedding. 2009. Bird or bat: comparing airframe design and flight performance. *Bioinspiration & Biomimetics* 4, 1 (2009), 015001. DOI:http://dx.doi.org/10.1088/1748-3182

Wanja Hofer, Daniel Lohmann, Fabian Scheler, and Wolfgang Schröder-Preikschat. 2009. Sloth: Threads as Interrupts. In *30th IEEE Real-Time Systems Symposium (RTSS 2009)*. IEEE, Washington, DC, 204–213. DOI:http://dx.doi.org/10.1109/RTSS.2009.18

Timo Hönig, Christopher Eibel, Rüdiger Kapitza, and Wolfgang Schröder-Preikschat. 2011. SEEP: Exploiting Symbolic Execution for Energy-aware Programming. In *23nd ACM Symposium on Operating Systems Principles (SOSP 2011), 4th Workshop on Power-Aware Computing and Systems (HotPower 2011)*. ACM, Cascais, Portugal, 4:1–4:5. DOI:http://dx.doi.org/10.1145/2039252.2039256

Timo Hönig, Heiko Janker, Christopher Eibel, Oliver Mihelic, Rüdiger Kapitza, and Wolfgang Schröder-Preikschat. 2014. Proactive Energy-Aware Programming with PEEK. In *USENIX Conference on Timely Results in Operating Systems (TRIOS 2014)*. Broomfield, CO, 1–14.

Philo Juang, Hidekazu Oki, Yong Wang, Margaret Martonosi, Li-Shiuan Peh, and Daniel Rubenstein. 2002. Energy-Efficient Computing for Wildlife Tracking: Design Tradeoffs and Early Experiences with ZebraNet. *ACM SIGOPS Operating Systems Review* 36, 5 (Dec. 2002), 96–107. DOI:http://dx.doi.org/10.1145/635508.605408

Raja Jurdak, Philipp Sommer, Branislav Kusy, Navinda Kottege, Christopher Crossman, Adam Mckeown, and David Westcott. 2013. Camazotz: Multimodal Activity-based GPS Sampling. In *12th ACM/IEEE International Symposium on Information Processing in Sensor Networks (IPSN 2013)*. ACM, Philadelphia, PA, 67–78. DOI:http://dx.doi.org/10.1145/2461381.2461393

Sukun Kim, R. Fonseca, and D. Culler. 2004. Reliable transfer on wireless sensor networks. In *1st IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON 2004)*. IEEE, Santa Clara, CA, 449–459. DOI:http://dx.doi.org/10.1109/SAHCN.2004.1381947

Andreas Köpke, Michael Swigulski, Karl Wessel, Daniel Willkomm, P.T. Klein Haneveld, Tom Parker, Otto Visser, Hermann Simon Lichte, and Stefan Valentin. 2008. Simulating Wireless and Mobile Networks in OMNeT++ – The MiXiM Vision. In *1st ACM/ICST International Conference on Simulation Tools and Techniques for Communications, Networks and Systems (SIMUTools 2008): 1st ACM/ICST International Workshop on OMNeT++ (OMNeT++ 2008)*. ACM, Marseille, France.

Jó Agila Bitsch Link, Gregor Fabritius, Muhammad Hamad Alizai, and Klaus Wehrle. 2010. BurrowView - seeing the world through the eyes of rats. In *8th IEEE International Conference on Pervasive Computing and Communications (PERCOM 2010), 2nd IEEE International Workshop on Information Quality and Quality of Service for Pervasive Computing (IQ2S 2010)*. IEEE, Mannheim, Germany, 56–61. DOI:http://dx.doi.org/10.1109/PERCOMW.2010.5470603

Michael Luby. 2002. LT Codes. In *43rd Symposium on Foundations of Computer Science (FOCS 2002)*. IEEE, Vancouver, BC, Canada, 271–280. DOI:http://dx.doi.org/10.1109/SFCS.2002.1181950

Michael G. Luby, Michael Mitzenmacher, M. Amin Shokrollahi, and Daniel A. Spielman. 2001. Efficient Erasure Correcting Codes. *IEEE Transactions on Information Theory* 47, 2 (2001), 569–584. DOI:http://dx.doi.org/10.1109/18.910575

Alan Mainwaring, Joseph Polastre, Robert Szewczyk, David Culler, and John Anderson. 2002. Wireless Sensor Networks for Habitat Monitoring. In *1st ACM Workshop on Wireless Sensor Networks and Applications (WSNA 2002)*. Atlanta, GA.

Margit Mutschlechner, Patrick Baldemaier, Philipp Handle, and Falko Dressler. 2013. Wireless in The Woods: Experimental Evaluation of IEEE 802.11a/b/g in Forested Environments. In *12. GI/ITG KuVS Fachgespräch Drahtlose Sensornetze (FGSN 2013)*. Cottbus, Germany, 5–8.

Margit Mutschlechner, Florian Klingler, Felix Erlacher, Florian Hagenauer, Marcel Kiessling, and Falko Dressler. 2014a. Reliable Communication using Erasure Codes for Monitoring Bats in the Wild. In *33rd IEEE Conference on Computer Communications (INFOCOM 2014), Student Activities*. IEEE, Toronto, Canada, 189–190. DOI:http://dx.doi.org/10.1109/INFCOMW.2014.6849219

Margit Mutschlechner, Bijun Li, Ruediger Kapitza, and Falko Dressler. 2014b. Using Erasure Codes to Overcome Reliability Issues in Energy-Constrained Sensor Networks. In *11th IEEE/IFIP Conference on Wireless On demand Network Systems and Services (WONS 2014)*. IEEE, Obergurgl, Austria, 41–48. DOI:http://dx.doi.org/10.1109/WONS.2014.6814720

Michael Noisternig. 2004. Tornado-Codes. http://notion.muelln-kommune.net/cgi/tornado-paper.ps.cgi. (2004).

H.D. Pfister and P.H. Siegel. 2008. Joint iterative decoding of LDPC codes for channels with memory and erasure noise. *IEEE Journal on Selected Areas in Communications* 26, 2 (Feb. 2008), 320–337. DOI:http://dx.doi.org/10.1109/JSAC.2008.080209

Irving S. Reed and Gustave Solomon. 1960. Polynomial Codes Over Certain Finite Fields. *Journal of the Society for Industrial & Applied Mathematics* 8, 2 (1960), 300–304. DOI:http://dx.doi.org/10.1137/0108018

Luigi Rizzo. 1997. Effective Erasure Codes for Reliable Computer Communication Protocols. *ACM SIGCOMM Computer Communication Review (CCR)* 27, 2 (April 1997), 24–36. DOI:http://dx.doi.org/10.1145/263876.263881

Bernd-Ulrich Rudolph, Alois Liegl, and Otto Von Helversen. 2009. Habitat Selection and Activity Patterns in the Greater Mouse-Eared Bat Myotis myotis. *Acta Chiropterologica* 11, 2 (2009), 351–361. DOI:http://dx.doi.org/10.3161/150811009X485585

Danilo Russo, Gareth Jones, and Raphaël Arlettaz. 2007. Echolocation and passive listening by foraging mouse-eared bats Myotis myotis and M. blythii. *Journal of Experimental Biology* 210, 1 (2007), 166–176. DOI:http://dx.doi.org/10.1242/jeb.02644

Christian Rutz, Zackory T. Burns, Richard James, Stefanie M.H. Ismar, John Burt, Brian Otis, Jayson Bowen, and James J.H. St Clair. 2012. Automated mapping of social networks in wild birds. *Current Biology* 22, 17 (2012), R669–R671. DOI:http://dx.doi.org/10.1016/j.cub.2012.06.037

Amin Shokrollahi. 2006. Raptor Codes. *IEEE Transactions on Information Theory* 52, 6 (June 2006), 2551–2567. DOI:http://dx.doi.org/10.1109/TIT.2006.874390

Reinald Skiba. 2003. *Europäische Fledermäuse: Kennzeichen, Echoortung und Detektoranwendung*. Westarp-Wissenschaften. 212 pages.

Philipp Sommer, Branislav Kusy, Adam McKeown, and Raja Jurdak. 2014. The Big Night Out: Experiences from Tracking Flying Foxes with Delay-Tolerant Wireless Networking. In *Real-World Wireless Sensor Networks*, Koen Langendoen, Wen Hu, Federico Ferrari, Marco Zimmerling, and Luca Mottola (Eds.). Lecture Notes in Electrical Engineering, Vol. 281. Springer International Publishing, 15–27. DOI:http://dx.doi.org/10.1007/978-3-319-03071-5_2

András Varga. 2001. The OMNeT++ Discrete Event Simulation System. In *European Simulation Multiconference (ESM 2001)*. Prague, Czech Republic.

G. M. Viswanathan, V. Afanasyev, S. V. Buldyrev, E. J. Murphy, P. A. Prince, and H. E. Stanley. 1996. Lévy flight search patterns of wandering albatrosses. *Nature* 381, 6581 (May 1996), 413–415. DOI:http://dx.doi.org/10.1038/381413a0

Mehmet C. Vuran and Ian F. Akyildiz. 2009. Error Control in Wireless Sensor Networks: A Cross Layer Analysis. *IEEE/ACM Transactions on Networking (TON)* 17, 4 (Aug. 2009), 1186–1199. DOI:http://dx.doi.org/10.1109/TNET.2008.2009971