All is not Lost: Understanding and Exploiting Packet Corruption in Outdoor Sensor Networks

Frederik Hermans, Hjalmar Wennerström, Liam McNamara, Christian Rohner, Per Gunningberg

Uppsala Universitet, Sweden

Abstract. During phases of transient connectivity, sensor nodes receive a substantial number of corrupt packets. These corrupt packets are generally discarded, losing the sent information and wasting the energy put into transmitting and receiving. Our analysis of one year's data from an outdoor sensor network deployment shows that packet corruption follows a distinct pattern that is observed on all links. We explain the pattern's core features by considering implementation aspects of low-cost 802.15.4 transceivers and independent transmission errors. Based on the insight into the corruption pattern, we propose a probabilistic approach to recover information about the original content of a corrupt packet. Our approach vastly reduces the uncertainty about the original content, as measured by a manifold reduction in entropy. We conclude that the practice of discarding all corrupt packets in an outdoor sensor network may be unnecessarily wasteful, given that a considerable amount of information can be extracted from them.

Keywords: wireless, transmission errors, packet corruption, outdoor sensor networks, robustness, 802.15.4

1 Introduction

Outdoor sensor networks experience significant variations in radio link performance over time [1, 2]. When links are in a transient state, they receive a large amount of corrupt packets, which are commonly discarded. Consequently, the information the sender intended to transmit is lost and has to be retransmitted. Therefore, corrupt packets incur a cost on the networks' limited energy budget for both transmitting and receiving the corrupt packet, and for retransmission.

We study corrupt packets from an 802.15.4-based outdoor deployment in a remote area. We find that corruption occurs to a non-negligible degree on intermediate links. It emerges that corruption follows a distinct, stable pattern that holds over various time scales and across links. We explain this pattern by considering an implementation aspect of low-cost 802.15.4 transceivers, the tie resolution strategy in coding, and a channel model in which errors occur independently. While corruption in packets has been studied recently in outdoor networks [3] and earlier in the case of interference [4, 5], we are the first to explain the occurence of the observed pattern.

Some earlier work has also addressed how to make use of corrupt packets. Apart from forward error correction, the approaches either selectively retransmit parts of a packet that are suspected to be corrupted [6, 7], or aim to reconstruct a correct packet from multiple corrupt packets [8].

We take a novel path to handling corrupt packets. We note that data in sensor networks is often inherently uncertain, e.g., due to limited accuracy of sensor readings. We therefore propose an approach that—rather than trying to exactly reconstruct a corrupt packet—probabilistically infers the packet's original content by exploiting the pattern in corruption. In combination with application knowledge, our approach enables recovery of information from corrupt packets. In contrast to earlier work, our approach does not need retransmissions of corrupt packets and hence does not incur an additional communication cost.

The evaluation of our approach shows that the uncertainty associated with a corrupt packet can be reduced significantly, as measured by an up to eight-fold reduction in entropy. We further validate our approach by applying it to data collected from a second deployment in another location, and find that it enables recovery of information from corrupt packets.

In summary, this paper makes the following core contributions:

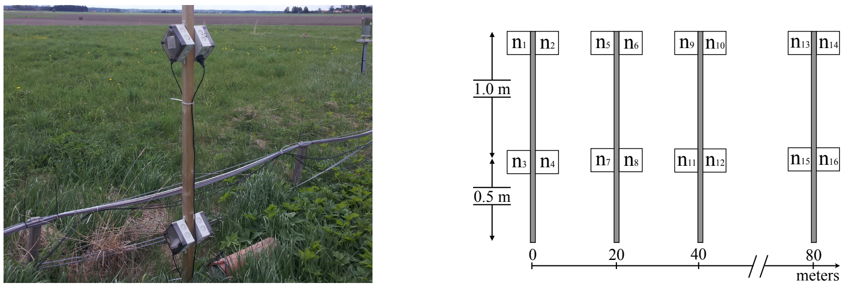
– By analyzing data from a long-term outdoor deployment, we describe the distinct pattern of how 802.15.4 packets are corrupted. Crucially, we can explain the pattern by considering implementation aspects of low-cost 802.15.4 transceivers and a simple radio channel model.
– Based on our insights, we describe an approach that probabilistically infers the original content of a corrupt packet. We evaluate this approach on a data set from a separate deployment, and find that it enables recovery by correctly assigning high probabilities to the original content. We also achieve a manifold reduction in the uncertainty associated with a corrupt packet.

To ensure that this paper is focused and self-contained, we have decided to leave out certain systems aspects, which we will address in future work.

The rest of the paper is organized as follows. We describe our deployment and data collection in Sec. 2, and briefly recap the IEEE 802.15.4 standard in Sec. 3. We analyze packet corruption in our deployment in Sec. 4, and describe our recovery approach in Sec. 5. Section 6 evaluates the approach, followed by a brief discussion of practical aspects in Sec. 7. We then survey related work in Sec. 8 and conclude the paper in Sec. 9.

2 Deployment and Data Collection

We deployed a sensor network at the outskirts of Uppsala, Sweden. The network is located in an open field with no trees or bushes in the surroundings, in a remote location that very few people have access to. The deployment is therefore not affected by man-made radio interference, e.g., from WiFi or Bluetooth.


n_1 n_2 n_5 n_6 n_9 n_{10} n_{13} n_{14}
1.0 m
n_3 n_4 n_7 n_8 n_{11} n_{12} n_{15} n_{16}
0.5 m
0
20
40
80
meters

(a) Pole with four sensor nodes (b) Deployment layout

Fig. 1: Outdoor deployment. Sensor nodes are labeled 1–16 in the right figure.

The network is comprised of 16 TelosB sensor nodes, which are equipped with 802.15.4-compatible CC2420 radio transceivers that operate in the 2.4 GHz ISM band [9]. The nodes are attached to four poles, with four nodes per pole (Fig. 1a). The poles are aligned along a straight line with a distance of 20 m between consecutive poles, as shown in Fig. 1b. On each pole, two nodes are mounted at 0.5 m above the ground and two nodes are mounted at 1.5 m.

The purpose of the network is to study radio links in 802.15.4 outdoor networks. Therefore, nodes take turns in sending 34-byte long probing packets every 500 ms. Whenever a node receives a packet, it logs the received packet content and the signal-to-noise ratio and Link Quality Indication (LQI) associated with the packet. Rather than discarding corrupt packets, nodes are programmed to also log corrupt packets. For power supply and log data collection, all nodes are connected via 5 m long RF-shielded USB cables to low-power Linux machines, which in turn are connected via Ethernet to a regular desktop PC that acts as a central experiment monitor.

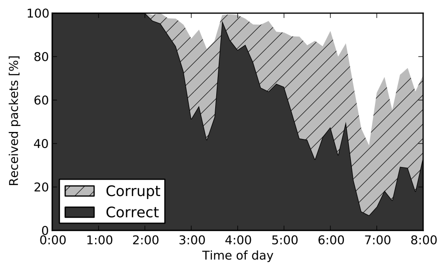
By analyzing the log files, which contain all sent and received packets (both correct and corrupt), we can determine which parts of a corrupt packet have suffered corruption. We use this information to analyze corruption in Sec. 4.

3 Recap of IEEE 802.15.4

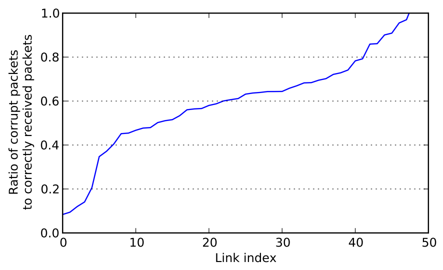
We briefly recapitulate the aspects of the IEEE 802.15.4 standard that are relevant to this paper. IEEE 802.15.4 is a standard for low-rate, low-power wireless communication [10], which has found wide-spread adoption in sensor networks.

A transmitted byte is represented by two four-bit *symbols*. 802.15.4 employs a direct sequence spread spectrum (DSSS) technique, in which each of the 16 possible symbols is represented by one *code word*. A code word, in turn, is represented by a 32-bit long pseudo-noise *chip sequence*.

We clarify the operation of 802.15.4 with an example. A sender wants to transmit a packet with n bytes of payload to a receiver. The sender translates each byte to two symbols. For each symbol, it determines the corresponding code word. The code words' chip sequences are then modulated onto a carrier frequency. The receiver demodulates the incoming chip sequences and matches



(a) Example of a deteriorating link



(b) Ratio of corrupt to correct packets

Fig. 2: The left figure exemplifies the amount of corrupt packets received for a specific link. The right figure shows that almost all intermediate links receive a substantial amount of corrupt packets.

them to the known code words. In this way, the receiver decodes $2n$ symbols, from which it can construct the n payload bytes.

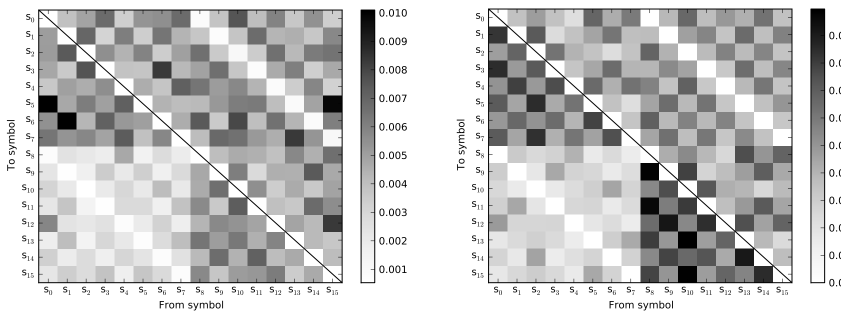
Synchronization is required to detect packet boundaries. A sender starts each packet with a predefined preamble, followed by a start frame delimiter and a length field. Upon decoding a preamble and start frame delimiter, a receiver knows that a packet is being transmitted. If the receiver fails to decode the preamble, the packet is lost.

Due to noise on the radio channel, a receiver's demodulated chip sequence may differ from the chip sequence transmitted by the sender. In this case, the incoming chip sequence is matched to the closest code word. DSSS thereby achieves resilience against noise, since there is a many-to-one mapping between chip sequences and code words. If sufficiently many chips are demodulated incorrectly [11], the receiver matches the incoming chip sequence to an incorrect code word, and hence decodes the wrong symbol. In this case, packet corruption occurs. To detect corruption, 802.15.4 packets end with a two-byte cyclic redundancy check (CRC) field. A receiver computes the CRC for the incoming packet and compares it against the received trailing CRC field. If they mismatch, the receiver knows that corruption has occurred.

4 Packet Corruption in an Outdoor Sensor Network

In this section, we analyze corrupt packets that were received by nodes in our deployment over the course of one year, from June 2012 to June 2013. We begin by briefly quantifying the amount of corruption occurring in the deployment. Then, we describe how corruption affects individual transmitted symbols, followed by a characterization of the effect of corruption on whole packets. Our analysis is focused on the regularities in corruption that enable the probabilistic recovery of information, as described in Sec. 5.

Because corrupt packets are usually discarded, the degree to which packet corruption occurs is unknown for most sensor networks. From our log data we



(a) Symbol mutations in data set (b) Symbol mutations in simulation

Fig. 3: Symbol mutations as observed in the data set and in simulation. Our simulation model produces the same core pattern as the empirical measurements.

observe that *intermediate links* (links that have a PRR between 10% and 90% [12, 13]) experience a substantial amount of packet corruption. Figure 2a shows a representative example of such an intermediate link. The depicted link initially has a high PRR, but deteriorates over time. As PRR falls, the amount of corrupt packets, indicated by the hatched gray area, grows.

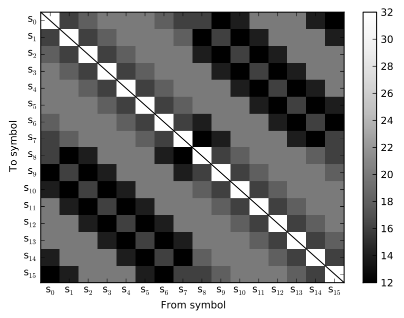
Next we look at the amount of corruption over all intermediate links from the duration of the deployment. We observe that about 80% of intermediate links have a ratio of corrupt packets to correctly received packets of at least 0.5. That is, for every two correctly received packets, they receive one corrupt packet on average. Figure 2b illustrates this ratio of corrupt packets to correctly received packets for intermediate links for a time span of two weeks in March 2013.

We conclude that packet corruption occurs at a non-negligible scale on intermediate links. Because intermediate links are the best candidates for improving network performance, this initial observation motivates us to understand packet corruption in more detail.

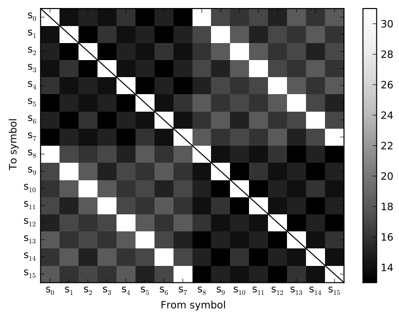
4.1 Corrupt Symbols

We now consider corruption at the finest level of granularity at which it can be observed in our deployment: the symbol level. If a node sends a packet containing a symbol s_i and due to corruption a receiver decodes the symbol incorrectly, which symbol s_j will the receiver likely decode?

Figure 3a shows how often each possible mutation $s_i \rightarrow s_j$ is observed over all links from the span of twelve months. The figure is a visual representation of the *mutation matrix*. An entry (j, i) of the mutation matrix denotes the frequency with which we observed a sent symbol s_i to be received as s_j. The matrix diagonal describes how often a symbol was decoded correctly. We omit the diagonal in the figure to focus on corruption. The darker the color in the figure, the higher the frequency. A distinct visual structure emerges in the figure, which leads us the following three observations:



(a) Hamming distances between code words of the 802.15.4 standard



(b) Hamming distances between MSK-transformed code words

Fig. 4: Hamming distances for 802.15.4 code words (left) and MSK-transformed code words (right). The MSK transformation explains observations 1 and 2.

Observation 1: Mutations are not uniformly distributed. If corruption occurs to a sent symbol, the received symbol depends on which symbol was sent. For example, if a node sends symbol s_0 which suffers corruption, the receiver most commonly decodes it as a s_5 (see column 0). Conversely, if a node receives a corrupt symbol as s_5, it is least common that s_{13} was sent (see row 5).

Observation 2: The most significant bit of a symbol is more stable than other bits. Note that the subdiagonal $(8, 0)$ to $(15, 7)$ and the superdiagonal $(0, 8)$ to $(7, 15)$ are lighter than the rest of the plot. For each column, the corresponding entry on these diagonals represents the symbol which differs from the sent symbol in only the most significant bit. For example, the sent symbol $s_0 = 0000_2$ is least commonly decoded as $s_8 = 1000_2$. This leads to the most significant bit of a symbol being more stable on average in our deployment.

Observation 3: Symbols s_0 to s_7 are more stable than other symbols. Consider the bottom left quadrant in Fig. 3a. It is significantly lighter than the other quadrants. This reflects the fact that in our data, the symbols s_0 to s_7 are unlikely to be decoded as s_8 to s_{15}. The converse is not true. Consequently, symbols s_0 to s_7 are less commonly corrupted than symbols s_8 to s_{15}.

The pattern shown in Fig. 3a represents mutation frequencies aggregated over all links over the whole time span of the deployment. We confirmed that the pattern also holds for individual links, and at various time scales. An independent research group has recently observed a similar pattern in an outdoor environment [3], which suggests the observations to be general.

To the best of our knowledge, no explanation has been offered so far as to why the pattern emerges. As Schmidt et al. point out [3], the pattern is surprising because it shows a negative correlation to the pairwise hamming distances of the code words defined in the 802.15.4 standard (see Fig. 4a). For example, the hamming distance between the code words for s_0 and s_8 is low, yet this mutation is among the least common in our deployment.

We attribute the first two observations to an implementation aspect of low-cost 802.15.4 transceivers. Rather than implementing an O-QPSK demodulator, as suggested in the 802.15.4 standard, many low-cost transceivers use an MSK demodulator instead [14]. While an MSK demodulator can correctly receive a chip sequence sent by an O-QPSK modulator, the received chip sequence will be transformed. Therefore, MSK-based 802.15.4 transceivers use a transformed set of code words to ensure compatibility with other 802.15.4 transceivers. The hamming distances between the transformed code words are shown in Fig. 4b.

Observation 1 can be explained by considering that each code word varies in its hamming distances to other code words. Therefore, the symbol that a corrupt chip sequence is decoded as depends on which symbol was sent.

Next, observation 2 follows directly from the observation that the MSK-transformed code word for symbol s_i has the highest hamming distance to the MSK-transformed code word for the symbol which differs from s_i only in the most significant bit. This is visualized by the light sub- and superdiagonals in Fig. 4b. Therefore, the most significant bit is more stable on average.

It remains to explain observation 3, which states that symbols s_0 to s_7 are more stable than the other symbols. This observation does not follow from the use of transformed code words, because code word distances are of course symmetric. We can explain the observation by considering how ties are resolved. A tie occurs if a received chip sequence matches two or more code words equally well. In this case, the transceiver must resolve the tie by choosing one of the matching code words. In a simple simulation, we found that if ties are resolved in a specific order[1], a pattern very similar to the empirically observed mutation matrix emerges (Fig. 3b). With the found order, a tie between two code words $s_{0\leq i\leq 7}$ and $s_{8\leq j\leq 15}$ will always be resolved in favor of the first symbol. Consequently, symbols s_0 to s_7 are more stable, as stated by observation 3.

Our simulation assumed one sender and one receiver, a fixed signal-to-noise ratio (SNR) at the receiver, and a channel model in which chip errors are independent, as would be expected in an additive white Gaussian noise channel, for example. Fixing the SNR implies a fixed chip error probability at the receiver.

We draw another useful conclusion from the similarity of the empirical and the simulated mutation matrix. The similarity suggests that the radio channel in the deployment can be modeled by a channel in which errors are independent, as assumed in our simulation. Note that differences in absolute values in Fig. 3a and Fig. 3b can be explained by considering that the simulated mutation matrix is based on a fixed SNR, whereas the empirical mutation matrix is based on packets received at various SNR levels. Nonetheless, the similarity holds.

In summary, we conclude that corruption follows a distinct pattern, which we attribute to MSK-transformed code words, the tie resolution strategy and a radio channel with independent chip errors.

[1] The order is $s_7, s_6, \ldots, s_0, s_{15}, s_{14}, ..., s_8$.

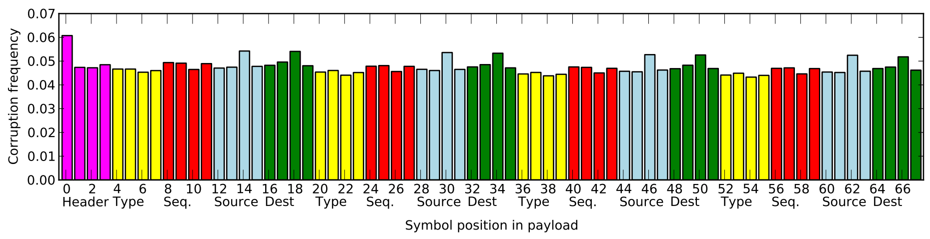


Fig. 5: Symbol error frequency over positions of the payload. Error frequencies are roughly uniform. Variations can be attributed to payload structure and content.

4.2 Errors in Packets

In the previous section, we considered the effect of corruption on individual symbols. We now shift our focus one layer up in the network stack to the link layer and consider how corruption affects whole packets, i.e., sequences of symbols.

Distribution of Errors How are symbol errors distributed within a packet? Figure 5 shows this distribution where for each position of the payload, the plot shows the frequency with which a symbol at this position was corrupt. The x-axis is annotated with the content of the payload. For example, the first four positions contain the packet header.

The error frequency is similar across all positions, ranging from 4.5% to 6%. Although the distribution is roughly uniform, there are notable deviations. First, the symbol at position 0 is most often corrupt. Second, there is a periodicity: the error frequencies for positions 4 to 19 are similar to the frequencies for positions 20 to 35, and so on. These deviations can be explained by the content of sent packets. For all packets sent in our deployment, position 0 always contains symbol s_8, which we know to be least stable. Furthermore, the payload of the probing packets sent in our deployment repeats itself after position 20, giving rise to the observed periodicity. Finally, because the sent packets contain structured rather than random content, some positions have slightly higher corruption frequencies than others. The observed deviations from uniformity are within range of the deviations we would expect due to the effects described in Sec. 4.1.

Correlation of Errors Are errors correlated? I.e., does an error at position x tell us something about whether an error occurred at position y? We computed pairwise correlations between all positions over all corrupt packets. The maximum absolute correlation between any two symbol positions is less than 0.09. Considering that a value of 0 indicates no correlation at all, we conclude that there are no notable correlations between errors at different positions. Therefore, symbol errors are independent from each other. This observation agrees with our assumption that the deployment's radio channel is well described by assuming independent chip errors.

Amount of Corruption in a Packet Finally, we quantify how many symbols in a corrupt packet are incorrect. Figure 6 shows a normalized histogram of the number of symbol errors per corrupt packet. The figure shows that most packets

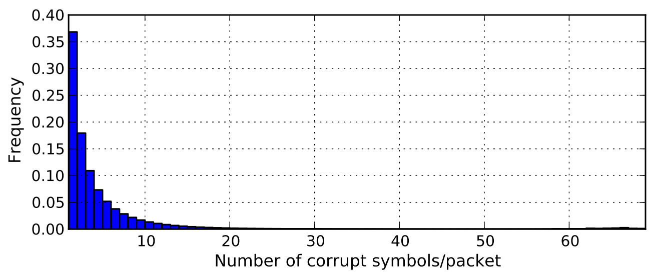


Fig. 6: Number of corrupt symbols per packet. Most suffer only little corruption.

have very few errors, and that the frequency of occurrence decreases with an increase in the amount of corruption.

5 Recovering Data from Corrupt Packets

We now describe how we use the observations from the previous section in an approach which for a given corrupt packet defines a probability distribution over the possible sent packets.

Computing a Probability Distribution over Possible Sent Packets We consider how to infer the likely sent data from a received corrupt packet. Our goal is to assign probabilities to the possible sent data, given the data in a corrupt packet. Recall from Sec. 3 that corruption occurs if sufficiently many chips in an incoming chip sequences are decoded incorrectly and hence the chip sequence is matched to the wrong code word. For the remainder of this analysis, we denote the probability of an individual chip in a received chip sequence being flipped as p_{chip}. For now, assume that we know p_{chip} for each received packet. We will revisit this assumption in the next section.

For a given value of p_{chip}, we can compute through simulation a corresponding mutation matrix $M^{p_{\text{chip}}}$. For example, the mutation matrix shown in Fig. 3b describes the mutation probabilities for $p_{\text{chip}} = 0.3$. Most importantly to our approach, the matrix rows describe the mutation probabilities for a received symbol. Note that the matrix main diagonal describes the probabilities of a symbol being received correctly.

For a given received packet, we now want to infer the first symbol of the sent packet. Let the first symbol of the received packet be s_j, and consider the case in which we know the packet to be corrupt because the CRC failed. By considering row j of $M^{p_{\text{chip}}}$, we can assign a probability to each possible sent symbol that could have led to the receiver decoding s_j. More specifically, the probability that the sent symbol s_i is decoded as the received symbol s_j is given by the entry $M^{p_{\text{chip}}}_{j,i}$ of the mutation matrix. We write $p(s_i|s_j) = M^{p_{\text{chip}}}_{j,i}$. For example, the probability $p(s_5|s_{13})$ that symbol s_5 was sent when s_{13} was received is given by $M^{p_{\text{chip}}}_{13,5}$. This reasoning holds for all position of the packet.

Because symbol errors are independent, we can readily assign probabilities to sequences of sent symbols. Assume, for example, that a receiver decoded the sequence of symbols $r = (s_{13}, s_3, s_0, s_{11})$. What is the probability that the actual sent symbols were $t = (s_5, s_3, s_1, s_{11})$? Due to independence, this probability is given by the product of the individual mutation probabilities:

$$\begin{aligned} p(t|r) &= p(s_5, s_3, s_1, s_{11}|s_{13}, s_3, s_0, s_{11}) \\ &= p(s_5|s_{13}) \cdot p(s_3|s_3) \cdot p(s_1|s_0) \cdot p(s_{11}|s_{11}) \\ &= M^{p_{\text{chip}}}_{13,5} \cdot M^{p_{\text{chip}}}_{3,3} \cdot M^{p_{\text{chip}}}_{0,1} \cdot M^{p_{\text{chip}}}_{11,11} \end{aligned}$$

In the manner we just outlined, a probability can be assigned to every possible sent symbol sequence for a given received symbol sequence and a given value of p_{chip}. This conceptually simple idea comprises our recovery approach. For each received, corrupt packet, we can compute a probability distribution over the possible sent packets. To compute the distribution, all we need to know is the chip error probability p_{chip} during packet reception.

To summarize, our approach determines a probability distribution over the possible sent data for given received data in a corrupt packet and a given p_{chip}. This concludes our description of recovery. We deliberately do not specify how the probability distribution is to be used by an application, because we believe that application knowledge should drive this process.

Estimating p_{chip} To assign probabilities to possible sent data, we need an estimate of the chip error probability p_{chip} for each corrupt packet. Unfortunately, low-cost transceivers do not provide such an estimate directly. Although there is a well-defined relationship between SNR and the chip error probability [15], we found the resolution of SNR reported by low-cost transceivers too low for a meaningful p_{chip} estimate.

To overcome this obstacle, we estimate p_{chip} for each packet by considering the LQI value associated with the packet. In the case of CC2420 transceivers, LQI is reflective of the correlation of an incoming chip sequence to the matched code word over the first eight symbols of a packet [9]. Therefore, we expect it to reflect the chip error rate. We construct a mapping from LQI values to chip error estimates as follows: for each LQI value l, we determine the empirically observed symbol error probability for symbol s_0. We then calculate the chip error probability p^l_{chip} that yields the same symbol error probability for s_0. We then construct a mapping from LQI to chip error probability by interpolating a 3rd degree spline through the resulting (l, p^l_{chip}) tuples. Our mapping is defined on LQI values in the range from 32 to 90, which covers 98.5% of all corrupt packets in our data set. We constrain the mapping to this range because we observe only very few corrupt packets with LQI less than 32 or higher than 90, and we therefore have little support to construct a mapping for these values.

While we do not expect our LQI to p_{chip} mapping to be perfect, we note that given the information that low-cost transceiver usually provide about the channel, it is difficult construct with a more well-defined estimate. We are confident that if transceivers were to provide high-resolution SNR measurements, a more exact estimator of chip error probabilities can be designed.

6 Evaluation

Our proposed approach defines a probability distribution over the possible sent data for given received data in a corrupt packet. We now address two questions pertaining to the resulting distributions. First, to what extent does the approach reduce uncertainty about the original content of a corrupt packet? Second, is the resulting distribution for a given corrupt packet meaningful? I.e., does it assign probabilities in a way such that the actual sent data has a high probability?

6.1 Reduction in Uncertainty

Let us address the first question of how much our approach reduces the uncertainty associated with a received, corrupt packet.

We consider the case in which a node sends a packet containing a 16-bit word t that we are interested in. This word could, for example, encode a sensor measurement, but for the sake of this analysis, we assume that we do not have any application-specific knowledge about the likely content t. We assume that a corrupt packet is received that contains the 16-bit word r. Due to the corruption, we do not know whether $r = t$ or not.

As a base case, assume that the corrupt packet is simply discarded. In this case, we know nothing about t. It could have taken any of the $2^{16} = 65{,}536$ possible values with equal probability. We measure the uncertainty associated with the discarded, corrupt packet by considering the entropy of the probability distribution over all possible words t' that could have been transmitted. There are 16 bits of entropy:

$$\begin{aligned} H_{\text{discard}} &= -\sum_{t'=0}^{65535} p(t') \log p(t') \\ &= -\sum_{t'} 2^{-16} \log 2^{-16} = 16. \end{aligned}$$

Next, we consider the case in which we do not discard the corrupt packet. The corrupt packet contains a word r, but we do not know if $r = t$. Using the approach described in Sec. 5, we can compute the probability $p(t'|r)$ for every possible 16-bit word t'. As in the case of the discarded packet, we can compute the entropy, which depends on the estimate of p_{chip} and on the received word r:

$$H_r = -\sum_{t'=0}^{65535} p(t'|r) \log p(t'|r).$$

Figure 7 depicts the entropy for both the base line case and our approach for different values of p_{chip}. The y-axis denotes the entropy. The x-axis relates to the received word r as described below.

In the base case, in which the corrupt packet is discarded, the entropy is 16 regardless of which word was received in the corrupt packet, and regardless of the chip error probability. With our approach, which assigns probabilities

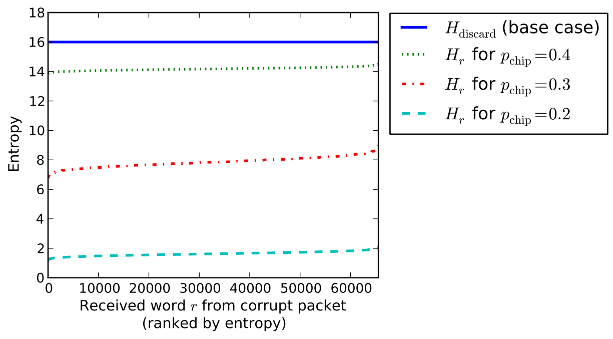


Fig. 7: Uncertainty associated with a corrupt packet in the case the packet is discarded, and when our proposed approach is used. Our approach significantly reduces the uncertainty, as measured by entropy.

the possible sent words by considering which word r was received in the corrupt packet, the entropy depends on r. This is an effect of our observation that symbol mutations are not uniform. For a given chip error probability p_{chip}, the entropies H_r are plotted in increasing order along the x-axis.

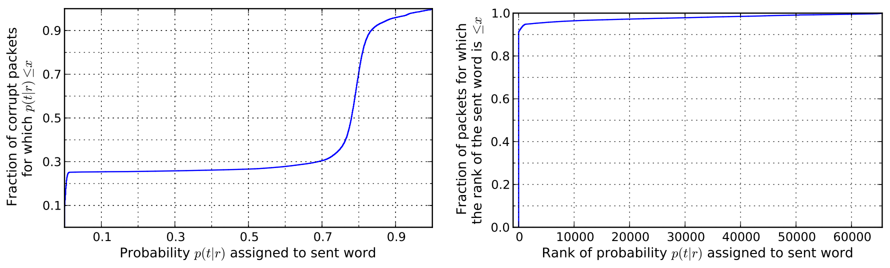
The figure makes it clear that our approach significantly reduces the entropy associated with a corrupt 16-bit word. In the case of a chip error probability of 0.2, the entropy is reduced to less than 2 bits—an eight-fold reduction of the entropy of the base case. For a higher chip error probability of 0.3, the entropy is still halved in comparison to the base case. In the case of an extreme chip error probability of 0.4, the entropy is reduced by two bits. However, for such a high chip error probability, most packets will be lost rather than corrupt because the preamble is likely to be corrupt as well. We therefore conclude that for realistic chip error probabilities of 0.2 to 0.3, our approach vastly reduces the uncertainty associated with a corrupt packet.

6.2 Evaluation of Probability Assignment

We have shown how our probability assignment reduces the uncertainty associated with a corrupt packet. It remains to show that the probability assignment is sensible, i.e., that there is a meaningful relationship between the probability assigned to possible sent words t' and the word t that was actually sent.

To address this question, we consider corrupt packets from a deployment different from the one that provided the data for the analysis in Sec. 4. Evaluating our approach on data from a different deployment increases our confidence in the generality of our findings, and helps understand whether our approach is strongly tied to the observations from our deployment in Uppsala. The other deployment is located in the Abisko national park in northern Sweden, which lies above the polar circle (latitude of $66^\circ\, 33'\, 44''$ N) in a climate that differs significantly from the climate in Uppsala. The Abisko deployment consists of 12 TelosB sensor nodes, and has a spatial layout similar to the Uppsala deployment. We consider corrupt packets from Abisko that were received during the first week of April 2013. We focus on this week because the deployment did not have any operational problems, such as failing nodes.

The data set of the Abisko deployment contains ca. 440,000 corrupt packets. We know the correct payload for each packet from our log data. We consider a



(a) Probabilities assigned to sent word (b) Rank of the sent word

Fig. 8: Our approach correctly assigns a high probability to the actual sent word, both in absolute (left) and relative (right) terms.

16-bit word t in these packets that describes the source address of the sender. We do not use any knowledge about the possible content of this word. For each corrupt packet, we estimate the chip error probability p_{chip} based on the packet's LQI measurement, as described earlier. We use the estimate to compute $p(t|r)$, which is the probability that is assigned to the sent word t when r was received. Clearly, it is desirable that a high probability be assigned to the sent word.

Figure 8a shows the empirical cumulative distribution function of the probability assigned to the sent word t for each corrupt packet. The x-axis shows the probability assigned to the sent word. The y-axis shows for how many of the packets this probability was below the corresponding x value. Note that for only 30 % of the corrupt packets, a probability of less than 0.7 is assigned to the sent word. For only 28 % of the corrupt packets, the probability assigned to the sent word is less than 0.5. It follows that for most corrupt packets, a high probability is assigned to the word that was actually sent. For these packets the probability assignment is sensible. However, a very low probability is assigned to the correct word for about 25% of the corrupt packets. Note that this does not imply that the assignment is wrong—in cases of a high chip error probability, there will be high uncertainty. High uncertainty means that the probability distribution will be more uniform across all possible sent words. The question thus is whether these low-probability assignments come from corrupt packets with high chip error probabilities. Before turning to this question, we conclude from Fig. 8a that for more than 70% of the corrupt packets, the probability assignment is sensible, because it assigns a high probability to the sent word.

We now order all possible sent words by decreasing order of assigned probability and determine the rank. E.g., the word t' that has been assigned the highest probability has rank one, the word with the second highest probability has rank two, etc. If two or more words have the same probability, they have the same rank. We are interested in the rank assigned to the sent word t.

The distribution of rank of the sent word is shown in Fig. 8b by an empirical cumulative distribution function. The figure shows that in 95% of the cases, the sent word is assigned a very high rank. I.e., the sent word is assigned a higher probability than most other possible candidates. We take this as an indication that even in cases where the highest probability is not assigned to the sent word, the sent word still takes a very high probability in comparison to other possible candidates. For the remaining 5% of corrupt packets, the rank is almost uniformly distributed up to the maximal rank of 65,536. We attribute this observation to misestimations of the chip error probability p_{chip}. Since we estimate p_{chip} from LQI, and LQI is only measured over the first eight symbols of a packet, it may by pure chance sometime misrepresent the chip error probability.

To summarize, we have shown in this section that for most corrupt packets, the sent word is assigned a high probability in comparison to other candidates. We conclude that the probability assignment we described in Sec. 5 indeed assigns probabilities in a meaningful manner. This observation suggests that our estimator of p_{chip} based on LQI is sufficiently accurate to enable recovery. Our approach thus enables sensor networks to infer the possible sent word corresponding to a corrupt packet.

7 Practical Considerations

We briefly discuss three aspects of practicality.

First, our approach determines a probability distribution over possible sent data. It does, by design, not produce a single value. When application knowledge about the likely content of a packet is available, this knowledge can be combined with our probability distribution to constrain the likely sent data even further. Application knowledge could be, for example, knowing the domain of a measured value from previous measurements. Such knowledge is often used to detect outliers, assess data quality, or handle missing data (e.g., see [11, 16, 17]). Such approaches are largely orthogonal to our proposal. Because the distributions computed by our approach are not centered around a single value, we believe that in combination with application knowledge, an even more exact inference of the content of corrupt packets is possible.

Second, a related question pertains to the complexity of our proposal. Note that the maths involved in determining the probability distribution is computationally very simple. Yet, for a received n symbol sequence, there are 16^n different possible sent values in the case of corruption. Enumerating all of them is infeasible for larger values of n. However, even for situations with moderately high chip error probabilities, many probabilities will be very close to zero. We envision that an application performing recovery will be interested in the top $k \ll 16^n$ possible sent sequences with the highest probabilities. These can be determined efficiently without enumerating all possible values. Therefore, we are confident that recovery can be performed in-network by nodes that are slightly more powerful than the TelosB-type nodes.

Third, the packets we analyzed in this paper were all sent and received by Texas Instruments CC2420 transceivers. Although this particular chip has a very high prevalence in academic research, the question arises of how well our findings translate to other 802.15.4 radio chipsets. In part, the observed pattern is an effect of the use of MSK demodulators, which are cheap to implement [18].

Therefore, they are common in low-cost transceivers. Consequently, we expect the pattern to hold for other transceiver, too. Note that because the pattern emerges even on short time scales, its presence in a particular radio chipset can be readily verified in an anechoic chamber.

8 Related Work

Wireless channels are inherently unstable [19, 12], causing errors in transmissions, and making mitigation strategies for these errors a wide field of research.

Schmidt et al. study corruption in an 802.15.4-based outdoor network and make observations similar to ours [3]. They point out that bit errors are not equally probable over all positions in the payload in 802.15.4 packets. They compare their empirical results to the expected values using code words as used with O-QPSK modulation. Han et al. identify patterns in the bit error probabilities of the payload in 802.11, which are not due to the channel conditions nor hardware-specific [20].

By using a software-defined radio, Wu et al. characterize the error patterns of individual 802.15.4 chip sequences in order to determine the channel conditions [21]. Similarly, Jamieson et al. implement a scheme in which they count the differences between the received and the known chip sequences to estimate the likelihood of a symbol being corrupt [6]. They then use this information, as part of a MAC protocol, to only re-transmit symbols that were likely corrupted. Dubois-Ferrière et al. combine successive alternating packets in order to infer the correct payload [8]. They show that this is feasible even when consecutive packets are broken, making the approach more robust than regular forward error correction. Hauer et al. propose to selectively retransmit parts of a packet during which there was a strong variation in received signal strength [7].

9 Conclusion

In this paper, we have described how corruption systematically affects symbols and packets in an outdoor 802.15.4 sensor network. We described a pattern in corruption that we attributed to the use of MSK demodulators, a specific tie resolution strategy when decoding, and a channel model with independent errors. These insights allowed us to formulate a novel probabilistic approach to recover information from corrupt packets. We showed that the approach reduces the uncertainty associated with a corrupt packet, and that it correctly assigns a high probability to the data that was actually sent. We will address systems aspects of our approach in future work and develop a concrete implementation of the proposed ideas. We specifically plan to investigate the trade-off between data quality and energy consumption, as well as the the relationship of our proposed recovery mechanism to other approaches such as forward error correction.

We conclude that patterns in packet corruption in outdoor sensor networks can be understood, and that information may be recovered from some corrupt packets. All is not lost when it comes to corrupt packets, and therefore discarding all of them is unnecessarily wasteful.

References

1. Lin, S., Zhang, J., Zhou, G., Gu, L., Stankovic, J.A., He, T.: ATPC: adaptive transmission power control for wireless sensor networks. ACM SenSys '06
2. Wennerström, H., Hermans, F., Rensfelt, O., Rohner, C., Nordén, L.A.: A Long-Term Study of Correlations between Meteorological Conditions and 802.15.4 Link Performance. IEEE SECON '13
3. Schmidt, F., Ceriotti, M., Wehrle, K.: Bit Error Distribution and Mutation Patterns of Corrupted Packets in Low-Power Wireless Networks. WiNTECH '13
4. Liang, C.J.M., Priyantha, N.B., Liu, J., Terzis, A.: Surviving Wi-Fi Interference in Low Power ZigBee Networks. ACM SenSys '10
5. Hermans, F., Rensfelt, O., Voigt, T., Ngai, E., Nordén, L.A., Gunningberg, P.: SoNIC: Classifying Interference in 802.15.4 Sensor Networks. IPSN '13
6. Jamieson, K., Balakrishnan, H.: PPR: partial packet recovery for wireless networks. ACM SIGCOMM '07
7. Hauer, J.H., Willig, A., Wolisz, A.: Mitigating the Effects of RF Interference through RSSI-Based Error Recovery. EWSN '10
8. Dubois-Ferrière, H., Estrin, D., Vetterli, M.: Packet combining in sensor networks. ACM SenSys '05
9. Texas Instruments Inc.: CC2420 - 2.4 GHz IEEE 802.15.4, ZigBee-ready RF Transceiver. `http://www.ti.com/lit/gpn/cc2420`
10. IEEE Computer Society: 802.15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs)
11. Wu, X., Liu, M.: In-situ soil moisture sensing: measurement scheduling and estimation using compressive sensing. ACM/IEEE IPSN '12
12. Srinivasan, K., Dutta, P., Tavakoli, A., Levis, P.: An empirical study of low-power wireless. ACM Trans. Sen. Netw. **6**(2) (March 2010)
13. Baccour, N., Koubâa, A., Mottola, L., Zúñiga, M.A., Youssef, H., Boano, C.A., Alves, M.: Radio link quality estimation in wireless sensor networks: A survey. ACM Trans. Sen. Netw. **8**(4) (September 2012)
14. Schmid, T.: GNU Radio 802.15.4 En- and Decoding. Technical report, Department of Electrical Engineering, University of California, Los Angeles '06
15. Xiong, F.: Digital Modulation Techniques. 2 edn. Artech House (April 2006)
16. Kong, L., Xia, M., Liu, X.Y., Wu, M.Y., Liu, X.: Data loss and reconstruction in sensor networks. IEEE INFOCOM '13
17. Hasenfratz, D., Saukh, O., Thiele, L.: Model-driven accuracy bounds for noisy sensor readings. IEEE DCOSS '13
18. Notor, J., Caviglia, A., Levy, G.: CMOS RFIC Architectures for IEEE 802.16.4 Networks. Technical report, Cadence Design Systems, Inc., '03
19. Zúñiga Zamalloa, M., Krishnamachari, B.: An analysis of unreliability and asymmetry in low-power wireless links. ACM Trans. Sen. Netw. **3**(2) (June 2007)
20. Han, B., Ji, L., Lee, S., Bhattacharjee, B., Miller, R.R.: Are All Bits Equal? Experimental Study of IEEE 802.11 Communication Bit Errors. IEEE/ACM Trans. Netw. **20**(6) (2012)
21. Wu, K., Tan, H., Ngan, H.L., Liu, Y., Ni, L.: Chip Error Pattern Analysis in IEEE 802.15.4. IEEE Trans. Mob. Comp. **11**(4) (2012)