Accurate Corruption Estimation in ZigBee under Cross-Technology Interference

Gonglong Chen, *Student Member, IEEE*, Wei Dong, *Member, IEEE*, Zhiwei Zhao, *Member, IEEE*, and Tao Gu, *Senior Member, IEEE*

Abstract—Cross-Technology Interference affects the operation of low-power ZigBee networks, especially under severe WiFi interference. Accurate corruption estimation is very important to improve the resilience of ZigBee transmissions. However, there are many limitations in existing approaches such as low accuracy, high overhead, and requirement of hardware modification. In this paper, we propose an accurate corruption estimation approach, AccuEst, which utilizes per-byte SINR (Signal-to-Interference-and-Noise Ratio) to detect corruption. We combine the use of pilot symbols with per-byte SINR to improve corruption detection accuracy, especially in highly noisy environments (i.e., noise and interference are at the same level). We extract pilot symbols by leveraging protocol signatures. In addition, we design an adaptive pilot instrumentation scheme to strike a good balance between accuracy and overhead. We implement AccuEst on the TinyOS 2.1.1/TelosB platform and evaluate its performance through extensive experiments. Results show that AccuEst improves corruption detection accuracy by 79.4 percent on average compared with state-of-the-art approach (i.e., CARE) in highly noisy environments. In addition, AccuEst reduces pilot overhead by 83.7 percent on average compared to the traditional pilot-based approach. We implement AccuEst in a coding-based transmission protocol, and results show that with AccuEst, the packet delivery ratio is improved by 22.1 percent on average.

Index Terms—Cross-technology interference, logistic regression, packet corruption, pilot symbol

1 Introduction

The recent years have witnessed the unprecedented proliferation of smart wireless devices. A number of radio technologies exist such as WiFi, Bluetooth and ZigBee for applications with different requirements in throughput, timeliness and energy-efficiency. Since these radio technologies operate on the same 2.4 GHz ISM band, it inevitably causes Cross-Technology Interference (CTI).

Prior studies have shown that ZigBee packets may suffer from severe corruption with WiFi interference [1], [2]. Many solutions have been proposed to improve the resilience of ZigBee transmission by first estimating corruptions and then recovering corrupted packets. On successfully identifying corruptions in a packet received, the receiver may be able to recover partial packet by requesting the retransmission of corruptions [3], enable whole packet recovery by combining the correct parts of multiple partial packets [4], [5], or recover more potential corrupted packets with the estimated corruptions (e.g., Reed Solomon code) when a packet is encoded with forward error correction (FEC) [6], [7] or rateless coding [8], [9]. All these methods rely heavily on the accuracy of corruption estimation.

Several corruption estimation methods exist in the literature [3], [10], [11], [12], [13], [14]. The PHY-based approach (e.g., PPR [3], AccuRate [10]) relies on the *detailed* PHY-layer information, e.g., Hamming distance between chip sequences or dispersion in the constellation space. Such an approach, albeit accurate, cannot work on COTS ZigBee devices since the detailed PHY-layer information is simply inaccessible. The pilot-based approach (e.g., ZipTx [11], LEAD [12]) relies on the *known* pilot symbols for coarse-grained BER (Bit Error Rate) estimation. Such an approach suffers from the inherent tradeoff between accuracy and overhead: if we instrument a small number of pilots, the accuracy will be low; otherwise, the packet overhead will be high. In general, this approach, when used alone, is *insufficient* to identify corruption in a packet. All the existing methods rely on hardware modification or incur large packet overhead.

Recently, in-packet RSSI sampling has accelerated much research interest [15], [16], [17], [18], [19], [20], [21]. In-packet RSSI sampling, working at the maximum sampling rate, provides fine-grained per-byte RSSI values for a packet. A key benefit of this technique is that it can be directly supported by COTS ZigBee devices without hardware modification. The fine-grained RSSI time series, provided by in-packet RSSI sampling, have been used in several works to classify interference or corruption estimation in the presence of WiFi interference. The *in-packet RSSI-based approach* such as REPE [15] and CARE [16] can work on COTS ZigBee devices with no packet overhead. However, they still suffer from *low accuracy*, especially in a common industry environment [22], [23] with high noise

- *G. Chen and W. Dong are with the College of Computer Science in Zhejiang University, Alibaba-Zhejiang University Joint Institute of Frontier Technologies, and Zhejiang Provincial Key Laboratory of Service Robot, Hangzhou 310027, China. E-mail: {desword, dongw}@zju.edu.cn.*
- *Z. Zhao is with the College of Computer Science and Engineering, University of Electronic Science and Technology of China, Sichuan 610051, China. E-mail: zzw@uestc.edu.cn.*
- *T. Gu is with the School of Computer Science and IT, RMIT University, Melbourne, VIC 3000, Australia. E-mail: tao.gu@rmit.edu.au.*

Manuscript received 3 Dec. 2017; revised 8 July 2018; accepted 25 Sept. 2018. Date of publication 16 Oct. 2018; date of current version 28 Aug. 2019. (Corresponding author: Wei Dong.)
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TMC.2018.2875744

(i.e., noise and interference are at the same level as we analyzed in Section 3.1). Achieving accurate corruption estimation in the in-packet RSSI-based approach is challenging. A straightforward idea is to sample noise level and explicitly mitigate the impact of noise power. However, WiFi interference is time-varying, hence it is not enough to distinguish noise from interference by sampling noise power. Our experimental study shows that existing in-packet RSSI-based approaches do not perform well in a highly noisy environment.

To address the above challenge, we propose a novel approach named AccuEst, to Accurately Estimate corruptions of ZigBee packets with WiFi interference in a highly noisy environment. We discover an interesting noise-resistance characteristic of link-layer pilot symbols which are known to both senders and receivers. This information can be used to indicate whether a symbol in a received packet is corrupted or not, and hence it can guide us to train a model to detect corruption. The PHY-layer information (e.g., per-byte RSSI values) offered by COTS ZigBee devices can be regarded as the input feature of the model. In particular, the per-byte RSSI values can capture the impact of interference (e.g., burstiness) in a *fine-grained* manner. We thus propose a new PHY-layer feature, per-byte SINR (Signal to Interference and Noise Ratio) deriving from per-byte RSSI values, to better indicate byte-level corruptions. In this way, we achieve better corruption estimation by combining *cross-layer* information.

While combining cross-layer information looks promising, network throughput may degrade since the link-layer information (e.g., pilot symbols) increases packet redundancy. Directly reducing the amount of the link-layer information, however, may decrease corruption estimation accuracy. It is thus challenging to ensure high corruption detection accuracy while minimizing the overhead. Therefore, we utilize a protocol signature based approach to extra pilot symbols from the received packet header. However, the number of extracted pilot symbols can not achieve relative high corruption estimation accuracy under certain interference patterns. We find that when the interference pattern (e.g., burstiness) is not obvious, the inferred per-byte SINR is highly possible to be interfered, resulting in the reduction of error estimation accuracy (as shown in Section 4.3). Therefore, in this case, we need to add more pilot symbols to quickly update the model to compensate for the interfered PHY-layer feature. Motivated by the above observations, we design an *interference pattern-aware* approach to strike a good balance between accuracy and overhead.

We implement AccuEst on TinyOS 2.1.1 with TelosB nodes and evaluate its performance in different environments. Our results show that AccuEst consistently achieves better performance than the state-of-the-art approach, i.e., CARE. Specifically, the improvement of corruption detection accuracy is obvious when interference level and noise level are close (i.e., 79.4 percent on average). AccuEst reduces pilot overhead by 83.7 percent on average compared to the traditional pilot-based approach while achieving the equivalent relative error of Symbol Error Rate (SER) estimation. To demonstrate the effectiveness of AccuEst in real scenarios, we implement AccuEst in a coding-based transmission protocol. The testbed results show that with AccuEst, the packet delivery ratio is improved by 22.1 percent on average.

The contributions are summarized as follows.

- We theoretically analyze and identify the limitations of existing in-packet RSSI-based corruption estimation approaches in highly noisy environments (i.e., noise and interference are at the same level).
- We propose a novel corruption estimation approach, which exploits cross-layer information and a learning-based model to achieve high accurate corruption prediction.
- We design an interference pattern-aware approach to minimize overhead while ensuring high corruption detection accuracy.
- We implement AccuEst on the TelosB platform with TinyOS 2.1.1 and evaluate its performance extensively. The results show that AccuEst significantly improves the corruption detection accuracy compared with the state-of-the-art approach in in highly noisy environments.

The rest of this paper is organized as follows. Section 2 introduces the related work. Section 3 presents the motivations. Section 4 shows the design of AccuEst. Section 5 presents the evaluation results. Section 6 discusses the generality of AccuEst, and finally, Section 7 concludes this paper and discusses future research directions.

2 Related Work

In this section, we discuss the related work. We first introduce existing corruption estimation (i.e., PHY-based approach which requires hardware modification, and pilot-based approach which requires known pilot symbols). We then present the in-packet RSSI sampling technique and discuss how the per-byte RSSI time series can benefit the upper-layer protocol design.

2.1 PHY-Based Approach

The detailed PHY-layer information provides accurate hints and it has been utilized by much prior works to identify erroneous bits [3], estimate BER [10], [14] or classify interference [5], [24].

PPR [3] implements an expanded PHY-layer interface called SoftPHY that provides a detailed PHY-layer hint about the PHY's confidence in each bit it decodes. Essentially, this confidence is derived from the Hamming distance between the actual received chip sequence and decoded chip sequence corresponding to a valid symbol. AccuRate [10] exploits the dispersion of the symbol constellation space to compute the optimal bit rate. Smaller dispersion means better link quality which is capable of supporting higher bit rates. By comparing these dispersions to the permissible dispersions at different bit rates, AccuRate derives the maximum rate to be used for packet transmission.

In addition to the above works which estimates corruptions, there are works targeting at accurately detecting the interference source. DoF [24] utilizes Fast Fourier Transform (FFT) to extract the repeating hidden patterns of different wireless protocols and then classifies interference sources according to the extracted patterns. CrossZig [5] exploits the variations in demodulated results of signals (i.e., soft values) for interference type detection (i.e., Intra- and

Cross-Technology Interference) and then enables an appropriate mechanism to recover the corrupted packet.

Different from the above approaches which require modification in hardware, our approach works directly on COTS ZigBee devices with no hardware modification.

2.2 Pilot-Based Approach

Pilot symbols/bits are the symbols/bits which are known before decoding and can be used to estimate BER [11], [13], [25] (Bit Error Rate) or assist channel decoding [12].

LEAD [12] extracts the fixed or highly biased header fields as the pilot bits and spreads the pilot bits over the whole packet to guide packet decoding. SmartPilot [13] further extracts more pilots from both detailed PHY-layer information and upper layer protocol headers to estimate BER and then picks a good data rate. However, they both require hardware modifications, limiting their usages on existing ZigBee devices.

ZipTx [11] evenly instruments the known pilot bits into a packet transmitted from sender. The receiver then estimates the BER of a packet based on the known pilot bits. The pilot-based approach does not require hardware modification. However, they only provide coarse-grained information which is insufficient to localize corruptions precisely. In addition, they incur relatively large packet overhead due to pilot symbols. To address this issue, we design an interference pattern-aware pilot instrumentation method to strike a good balance between accuracy and overhead.

2.3 In-Packet RSSI Approach

In-packet RSSI sampling has been recently proposed to provide fine-grained per-byte RSSI values for a packet, and it has accelerated a lot of interesting works [15], [16], [17], [18], [19], [20], [21], [26].

SoNIC [20] utilizes the key insight that different interferers disrupt individual 802.15.4 packets in different ways that can be detected by sensor nodes. Then the distinct patterns, e.g., variances of in-packet RSSI series, link quality indication and etc., can be used to classify different interference sources (e.g., Bluetooth and WiFi). Different from SoNIC, TIIM [19] skips the classification step and directly builds a decision tree to learn under which interference patterns a particular mitigation scheme empirically achieves the best performance. Smoggy-Link [21] maintains a link model to trace the relationship between interference and link quality of sender's outbound links. With such a link model, Smoggy-Link can obtain fine-grained spatiotemporal link information to perform adaptive link selection and transmission scheduling.

Song [27] et al. calculates the SINR from in-packet RSSI series to infer link correlations. Although AccuEst adopts the approach of calculating per-byte SINR, however it is only one of the components in extracting PHY-layer information. The core components of AccuEst are pilot symbol extraction, cross-layer information combination and interference pattern quantification. The performance of AccuEst is related to all above components, which are very different from [27] that relies heavily on the SINR calculation to estimate the link correlation.

REPE [15] utilizes the observation that RF interference typically manifests as an additive increase in RSSI [28]. It samples the RSSI values per symbol with a high-resolution hardware timer (i.e., 62.5 kHz). By calculating the difference between RSSI[i] (i.e., the combination of ZigBee signal and WiFi interference) and $RSSI_{base}$ (i.e., ZigBee signal strength without interference), REPE utilizes a single threshold-based approach to detect incorrect symbols. CARE [16] also exploits the in-packet RSSI time series to compute the corruption level of a packet. Then an adaptive coding scheme which is based on the corruption level is designed to retransmit the redundancy information.

However, as shown in Section 3.1, the difference between RSSI[i] and $RSSI_{base}$ suffers inevitable errors in a highly noisy environment (i.e., noise and interference are at the same level), and hence it degrades corruption detection accuracy. Different from REPE and CARE, our approach introduces a more accurate indicator per-byte SINR from the per-byte RSSI time series to detect corruption. In addition, we use the link-layer information (pilot symbols) to further improve corruption detection accuracy.

TALENT [29] combines cross-layer information to improve the link estimation accuracy. However, AccuEst extracts the byte-level features (e.g., per-byte SINR) and pilot symbols as the PHY-layer and the link layer information respectively, while TALENT only extracts the packet-level features (e.g., per-packet RSSI) and the packet reception ratio. Different from TALENT that predicts the PRR only based on packet-level information, AccuEst quantifies interference patterns using byte-level information, and adapts the amount of the extracted features to observed interference patterns. In this way, high accuracy and network throughput under various interference scenarios are achieved.

3 Motivation

3.1 Limitation of In-Packet RSSI Approach

In-packet RSSI values have been used in prior works such as CARE and REPE for corruption estimation. Given an array of RSSI (Received Signal Strength Indicator) values, RSSI[n], which corresponds to n bytes in a received packet. Each element in the array RSSI[i] ($1 \leq i \leq n$) indicates the RSSI in dBm for the ith byte in the received packet.

Both CARE and REPE detect corrupted bytes by using the difference between RSSI[i] and $\text{RSSI}_{base} = \min_i$ (RSSI[i]) as an indicator. RSSI_{base} roughly represents the ZigBee signal strength without interference, while RSSI[i] represents the combination of ZigBee signal and WiFi interference. If the RSSI difference, ΔRSSI[i], exceeds a threshold, it is highly probable that there exists a high interference and the corresponding byte is corrupted. We argue that ΔRSSI[i] may not be a robust indicator in some circumstances.

Before we perform a detailed analysis, we first introduce the following notations and formulas:

- The received power for the ith byte is denoted as $P_{mW}[i]$ which can be split into three components: $P^S_{mW}[i]$, $P^N_{mW}[i]$, $P^I_{mW}[i]$, representing the powers of signal, noise, and interference, respectively. We assume that the powers are additive [28], i.e., $P_{mW}[i] = P^S_{mW}[i] + P^N_{mW}[i] + P^I_{mW}[i]$.
- The power in mW (P_{mW}) can also be expressed in dBm (P_{dBm}) (and vice versa) by the following formula:

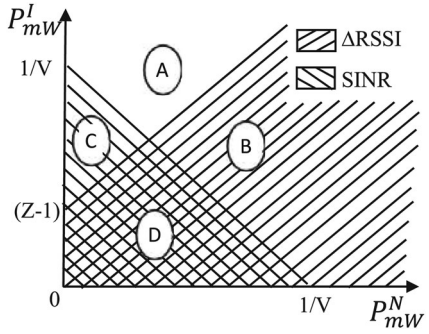


Fig. 1. Corruption detection using different indicators.

$$P_{dBm} = 10 \log_{10} P_{mW}. \tag{1}$$

ΔRSSI[i] as an Indicator. Based on the above notations, we can compute ΔRSSI[i] as follows:

$$\begin{aligned} \Delta RSSI[i] &= RSSI[i] - RSSI_{base} \\ &= 10 \log_{10}(P^S_{mW}[i] + P^N_{mW}[i] + P^I_{mW}[i]) \\ &\quad - 10 \log_{10}(P^S_{mW}[i] + P^N_{mW}[i]) \\ &= 10 \log_{10}\left(\frac{P^S_{mW}[i] + P^N_{mW}[i] + P^I_{mW}[i]}{P^S_{mW}[i] + P^N_{mW}[i]}\right) \\ &= 10 \log_{10}\left(1 + \frac{P^I_{mW}[i]}{P^S_{mW}[i] + P^N_{mW}[i]}\right). \end{aligned} \tag{2}$$

In both CARE and REPE, the byte corruption probability Pr_e has positive correlation with ΔRSSI[i], i.e., Pr_e increases when ΔRSSI[i] increases.

Standard Indicator. In reality, a standard indicator for *correct* transmission is the signal to interference and noise ratio, denoted as SINR, which can be computed as follows:

$$SINR_{dB}[i] = 10 \log_{10}\left(\frac{P^S_{mW}[i]}{P^I_{mW}[i] + P^N_{mW}[i]}\right). \tag{3}$$

The byte corruption probability Pr_e has negative correlation with $SINR_{dB}[i]$.

Analysis. When $P^N_{mW}[i] = 0$, the first indicator correctly identifies corruptions since Pr_e is large when ΔRSSI[i] is large (high interference). However, when $P^N_{mW}[i]$ increases to a large value comparable to $P^S_{mW}[i]$ (or $P^S_{mW}[i]$ decreases to a small value comparable to $P^N_{mW}[i]$), there will be inconsistency between the two indicators. Supposing $P^N_{mW}[i]$ increases to a large value, the ΔRSSI[i] will regard the byte as correct since large noise makes this indicator small. On the other hand, the standard SINR indicator will regard the byte as erroneous since large noise makes SINR small.

To better understand the above description, without loss of generality, we assume $P^S_{mW}[i]$ as 1 mw and simplify the standard indicator and the ΔRSSI[i] indicator to their antilogarithm part. Then, according to the standard indicator, we regard the byte as correct when the simplified standard indicator is larger than the threshold V

$$\frac{1}{P^I_{mW}[i] + P^N_{mW}[i]} > V. \tag{4}$$

According to the ΔRSSI[i] indicator, we regard the byte as correct when

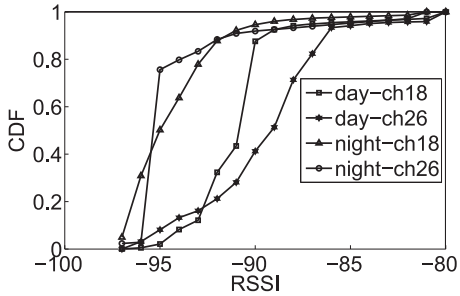


Fig. 2. Noise level in a typical office.

$$1 + \frac{P^I_{mW}[i]}{1 + P^N_{mW}[i]} < Z. \tag{5}$$

Where Z is the threshold for determining the correct byte. We plot Fig. 1 to clearly see the inconsistency. The quadrant is split into four regions by above two conditions. Regions C and D satisfy the condition (4), while Regions D and B satisfy the condition (5).

When $P^N_{mW}[i]$ increases to a large value comparable to $P^I_{mW}[i]$ (e.g., in Region B), the ΔRSSI[i] indicator classifies the byte as correct since large noise makes this indicator small, however, the standard indicator SINR will classify the byte as erroneous since large noise makes SINR small. Similarly, for Region C, since the ratio of $P^I_{mW}[i]$ and $P^N_{mW}[i]$ is large but the sum of them is small, we would make two opposite detection results using the ΔRSSI[i] indicator and the standard indicator SINR.

It is worth noting that the inconsistency in Region C can be eliminated by selecting two appropriate thresholds Z and V. However, since $Z - 1$ and $1/V$ are both larger than 0, the inconsistency between the ΔRSSI[i] indicator and the standard indicator SINR always exists in Region B. Therefore, the ΔRSSI[i] indicator cannot identify corruptions when noise and interference are at the same level.

3.2 Measurement Study in Practical Scenarios

Existing works [30] have shown that in industrial environments the noise level is as high as −86 dBm on average, varying from −92 dBm to −80 dBm. It is the common case for industrial scenarios, e.g., the indoor power control room. The high noise level mainly comes from the electromagnetic interference generated by the equipments in industrial scenarios [30], e.g., voltage transformer and power generator.

We also conduct measurement experiments in a typical office. The office is with a size of 15 m × 7 m, and consists of five servers, 45 personal computers and 42 laptops. We randomly place five TelosB nodes in the office to sample the noise level at 62.5 kHz. Note that we disable wireless protocols on the 2.4 GHz (e.g., WiFi and BLE) to eliminate the impact of high power Cross-Technology interferers. The noise level is measured on different channels (e.g., channel 18 and channel 26 for ZigBee) for both daytime and night. The results are merged over five TelosB nodes.

Fig. 2 presents the CDF of noise level in the office. We can find that during the daytime the noise level is −90 dBm on average over ZigBee channel 18 and 26, varying from −96 dBm to −80 dBm. The noise level during the daytime is a little bit higher than the night (e.g., −93 dBm on average, varying from −96 dBm to −85 dBm). The reason is when

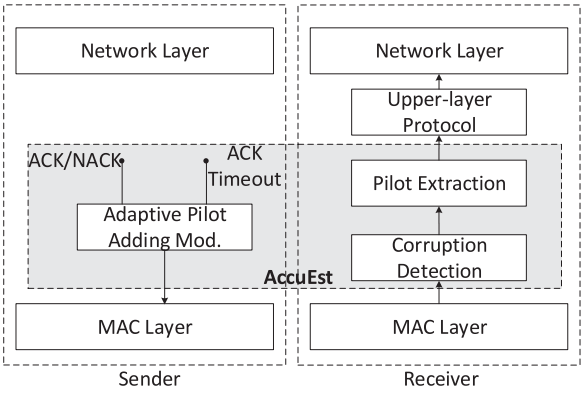


Fig. 3. The AccuEst architecture.

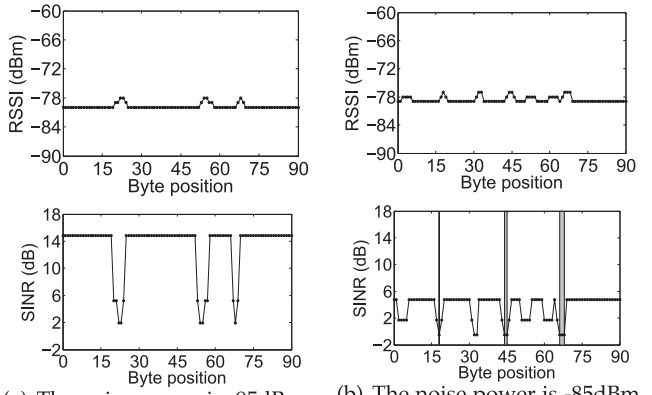


Fig. 4. illustrative examples of corruption detection under different noise power using the standard indicator and the ΔRSSI indicator, respectively.

more computers begin to operate, there will be more electromagnetic interference to the ZigBee devices.

From the above experiments, we conclude that for typical industry and office scenarios, the noise level is as high as −86 dBm and −90 dBm, which can easily reach the same level as the interference (e.g., when the interference is relatively low). As analyzed in Section 3.1, previous approaches suffer from low accuracy when the noise level is at the same level as the interference level. These findings motivate us to design a better error estimation approach.

4 Design

We describe the key idea in our approach as follows. We combine the pilot symbols with the accurate SINR indicator aiming to significantly improve the corruption detection accuracy, especially when noise and interference are at the same level. To achieve this, we first extract useful features from cross-layer information in a packet, i.e., in-packet RSSI time series from the PHY-layer and pilot symbols from the link-layer, to build a regression-based model. We then automatically train this model using known pilot symbols.

Fig. 3 shows the overall architecture of AccuEst. It sits between the MAC layer and the network layer. At the sender side, AccuEst instruments the pilot symbols evenly into the packets. The number of pilot symbols is computed according to the incoming events. (1) When the sender receives ACK/NACK packets carrying the information of the needed number of pilots, AccuEst instrument few extra pilots to the end of the packet header. (2) When the ACK timer times out, AccuEst delivers the event to the upper-layer protocol. At the receiver side, when a packet is received, the receiver needs to complete two tasks: (1) if this packet is corrupted, the receiver detects corruptions and deliveries the results to the upper layer protocol. (2) No matter this packet is corrupted or not, the receiver calculates the needed number of extra pilots and transmits ACK/NACK packets that carries the pilots number information to the sender. Note that comparing with our previous work AccuEst(ICDCS 17) [31], AccuEst can extra pilots from the received packet header, thus only a small number of extra pilots are needed if necessary.

Implementing the above idea has the following three challenges:

- Which features are essential to corruption detection?
- How to build the model and automatically train the model?
- How to design an adaptive approach to instrument pilots?

In the rest of this section, we detail the design of AccuEst to address above challenges.

4.1 Feature Extraction

We first introduce the information that can be obtained directly from the PHY-layer and the link-layer, and then show how to extract features that are most relevant to corruption detection.

1) PHY-layer Feature

Per-packet RSSI. It denotes the average RSSI value for the first eight symbols in a received packet. It only reflects the signal strength of the packet header, therefore it has limited capacity to detect corruptions in the whole packet.

Per-packet LQI. The link quality indication (LQI) is a characterisation of the quality of a received packet. CC2420 provides an average correlation value based on the first eight symbols to denote LQI. LQI has been used to detect the sudden changes in the packet header caused by interferers [20]. However, LQI alone provides limited information for determining erroneous bytes in the rest of the packet.

In-packet RSSI Time Series. We modify the radio driver in TinyOS 2.1.1 to sample RSSI at a rate of about one sample/byte. Our modified driver starts sampling RSSI whenever a Start Frame Delimiter (SFD) interrupt signals an incoming packet, and it keeps sampling until the last byte of the packet is received.

Fig. 4 presents the examples of detecting corruptions using the standard indicator and the ΔRSSI indicator. For the standard indicator, we regard the ith byte as correct when SINR[i] of the ith byte is larger than 1. For the ΔRSSI indicator, we use the threshold in CARE that when ΔRSSI[i] of the ith byte is lower than 2, we determine the ith byte as correct. When the interference and noise level are relatively low (i.e., as shown in Fig. 4a), there are no corrupted bytes and both the standard indicator and the ΔRSSI indicator can accurately identify the correct bytes. When the noise power increases from −95 dBm to −85 dBm, the packet is corrupted as shown in Fig. 4b. The grey region denotes the erroneous bytes that are detected correctly using the corresponding indicator. We cannot identify any erroneous bytes using the ΔRSSI indicator while most of the erroneous bytes are correctly detected using the standard indicator.

Therefore, we carefully select the per-byte SINR as our indicator. To infer SINR[i] of the ith byte, besides the in-packet RSSI time series, AccuEst samples the noise power as soon as the link turns idle after packet reception (i.e., $RSSI_n$). Let P^N_{mW} and P^S_{mW} denote the noise power and the signal power, respectively. The interference power of the ith byte is $P^I_{mW}[i]$, then SINR[i] can be computed as follows.

$$\begin{aligned} P^N_{mW} &= 10^{RSSI_n/10} \\ P^S_{mW} &= 10^{RSSI_{base}/10} - P^N_{mW} \\ P^I_{mW}[i] &= 10^{RSSI[i]/10} - P^S_{mW} - P^N_{mW} \\ SINR[i] &= 10\ log_{10} \frac{P^S_{mW}}{P^N_{mW} + P^I_{mW}[i]}. \end{aligned} \tag{6}$$

Where $RSSI_{base}$ is the minimum RSSI value during packet reception.

In summary, the PHY-layer feature vector $\boldsymbol{X_i}$ of the ith byte is expressed as follows.

$$\boldsymbol{X_i} = [SINR_i, LQI, RSSI_{pkt}]. \tag{7}$$

2) Link-layer Feature

Pilot Symbols. Pilot symbols are the symbols which are known before decoding. They provide a noise-resistance information about whether the symbol is correct or not in a received packet. In the rest of this paper, we will use *pilot symbols* and *pilots* interchangeably. According to the definition of 802.15.4 protocol, there are link invariant protocol data in the packet header. These data are likely to take a fixed value in a window of packets, which is referred as symbol bias or bit bias [12]. By carefully selecting a packet window size k, we can predict the value of some specific symbols and thus extract pilot symbols at the receiver side.

To extract all possible pilot symbols (e.g., four bits/symbol in 802.15.4 [32]), we define the bit bias according to [12] and then determine pilot symbols based on extracted biased bits. Given a link, we use $\boldsymbol{P}_{m,n}$ to represent a window of its packets.

$$\boldsymbol{P}_{m,n} = \begin{Bmatrix} b_{1,1} & \dots & b_{1,n} \\ \dots & \dots & \dots \\ b_{m,1} & \dots & b_{m,n} \end{Bmatrix}. \tag{8}$$

Where $b_{i,j}$ means the jth bit for the ith packet. There are m packets in the current window and n bits in a packet. The bit bias can be calculated as follows:

$$\alpha_i(\boldsymbol{P}_{m,n}) = 2 \times \mid \frac{\sum_{k=1}^{m} b_{k,i}}{m} - \frac{1}{2} \mid. \tag{9}$$

If bit i is fixed to the value 0 or 1 within m packets, $\alpha_i(\boldsymbol{P}_{m,n}) = 1$.

Based on above definition, we validate the biased bits using the collected trace from real-life 802.15.4 wireless networks. The collected trace is from the CC2420 links with different operation channel settings, i.e., channel 11 for WiFi and channel 15, 21 (overlapped), 26 for CC2420 radio. For each link in the above settings, we measure the bit bias using the first 300 correct packets. Due the similar results across difference link settings, we plot the averaged results as shown in Fig. 5. Among the 96 bits studied, 40 are fixed.

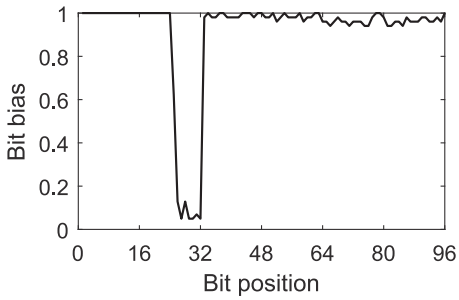


Fig. 5. Bit bias measured (802.15.4) with 300 packets.

A key component to classify biased bits is determining the observation window size, and we utilize the approach from [12]. After detecting biased bits, we still need to extract biased symbols for the model training.

To extract as many as possible pilot symbols, we classify symbols into three classes: biased symbols, intermediate biased symbols and unbiased symbols. Given the symbol size s (e.g., s = 4 bits in 802.15.4) and packet length L. We define the symbol bias as $V_j = \sum_{i=j\times s+1}^{s} \alpha_i$, $j = 0, \dots, L-1$. The symbol type T_j for the jth symbol can be classified as: 1) biased, when $V_j = s$; 2) intermediate, when $0 < V_j < s$; 3) unbiased, $V_j = 0$.

We directly regard the biased symbols as the link layer information. For the intermediate symbols, only when the intermediate symbols are determined as corrupted we can then treat them as link layer information. Combining the link-layer information and the PHY-layer feature makes it possible to train our model (as we shown in Section 4.2).

4.2 Combination of Cross-Layer Information

Suppose we have obtained the L pilot symbols from the received packet. Then our goal is: given the link-layer pilot symbols and the PHY-layer feature in a sliding window with size W, train and automatically update a model to determine the corruption probability of the bytes. Formally, the jth training set can be expressed as follows.

$$\boldsymbol{TrainS_j} = [\boldsymbol{PKT_j}, \boldsymbol{PKT_{j-1}}, \dots, \boldsymbol{PKT_{j-W+1}}]. \tag{10}$$

Where $\boldsymbol{PKT_j}$ is comprised of the PHY-layer feature as follows.

$$\boldsymbol{PKT_j} = [\boldsymbol{X^j_1}, \boldsymbol{X^j_2}, \dots, \boldsymbol{X^j_L}]. \tag{11}$$

Where $\boldsymbol{X^j_i}$ denotes the ith pilot PHY-layer feature of the jth packet (i.e., $SINR^j_i, LQI^j, RSSI^j_{pkt}$). In order to train the corruption detection model, we have the following label for the ith pilot symbol in jth packet.

$$P(Y = 1|\boldsymbol{X^j_i}) = \begin{cases} 0, correct \\ 1, erroneous. \end{cases} \tag{12}$$

Where $Y = 1$ denotes the byte is erroneous. If the pilot symbol is erroneous then we set $P(*) = 1$, meaning that the error probability of the byte is 1 and vice visa.

The corruption detection model should be lightweight to run on resource-constrainted sensor node. Therefore, we utilize the Logistic Regression (LR) to detect corrupted bytes. LR has been utilized by many prior works [29], [33] and it is easy to be implemented on sensor nodes.

TABLE 1
Four Scenarios Considering Interference Power (PAPR) and Average Number of Consecutive Corrupted Symbols (Bursty Level)

Scenarios	PAPR	Bursty Level
HIHB (Nearby video streaming in office)	≥ 3.2	10∼16
HILB (Nearby browser in office)	$\geq$3.2	$\leq$10
LIHB (Faraway video streaming in office)	1∼3.2	10∼16
LILB (Faraway browser in office)	1∼3.2	$\leq$10

H(High), L(Low), I(Interference), and B(Bursty).

We then apply stochastic gradient descent (SGD) to update parameters. SGD is an online algorithm that operates by repetitively drawing a fresh random sample and adjusting the weights on the basis of this single sample only [29]. The learning rate is set to 0.01 in our evaluation.

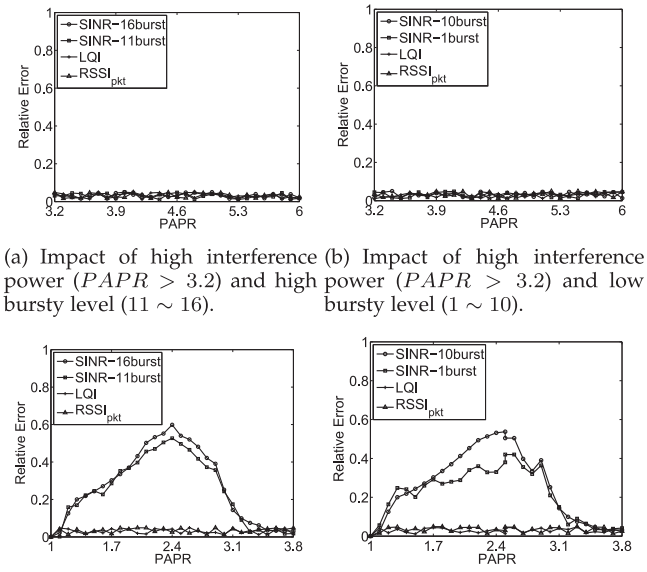
4.3 Adaptive Pilot Instrumentation

We now further improve corruption detection accuracy by combining the pilot symbols with the PHY-layer feature. However, instrumenting pilots would also degrade network throughput. Therefore, how to minimize the number of instrumented pilot symbols while keeping high accuracy of corruption detection is a challenging issue.

To better understand the benefits of combining the pilot symbols with the PHY-layer features, we first conduct an experimental study to show the impact of different interference patterns on the accuracy of inferred PHY-layer features. The details of experimental settings are similar to that in Section 5, except that an extra node is bounded with the receiver node and is enabled always-on RSSI sampling to obtain the ground truth of PHY-layer features. Our approach can detect the potential corrupted symbols, which CARE cannot identify, despite of any noise power as shown in Section 4.1. We thus only focus on four different interference power and traffic patterns (as shown in Table 1). The noise power is generated (i.e., -90 dBm) using USRP [34]. Before analyzing experimental results, we introduce the following features to quantify interference patterns:

PAPR (Peak-to-Average Power Ratio). It is a common measurement for the fluctuation of signal power and can be used to distinguish different PHY modulation techniques. We apply PAPR to analyze the WiFi interference power level. As shown in previous studies [17], 802.11g/n have a large PAPR ($\geq$1.9) while ZigBee has a relatively small PAPR ($\leq$1.3) because it employs the single-carrier modulation technique. The PAPR can be derived according to the existing work [17] with the normalized RSSI sequence of an N-byte packet $nRSSI$.

Bursty Level. Error burst means a sequence of corrupted symbols that may contain subsequences of at most four consecutive correct symbols in a packet. Prior work has observed that 802.15.4 corruptions under WiFi interference are highly bursty and the bursty density is utilized to classify different interference type [1], [20], [35]. We use a threshold-based approach to identify the start and the end points of each burst segment. The threshold thd is set to 2 dB according to [20]. Given the RSSI series $RSSI[i]$, the sets of start (S) and end (E) position can be expressed as follows.



(a) Impact of high interference power ($PAPR > 3.2$) and high bursty level ($11 \sim 16$).
(b) Impact of high interference power ($PAPR > 3.2$) and low bursty level ($1 \sim 10$).

(c) Impact of low interference power ($1 < PAPR < 3.2$) and high bursty level ($11 \sim 16$).
(d) Impact of low interference power ($1 < PAPR < 3.2$) and low bursty level ($1 \sim 10$).

Fig. 6. Impact of interference power (PAPR) and average number of consecutive corrupted symbols (bursty level) on the relative error of inferred PHY-layer features.

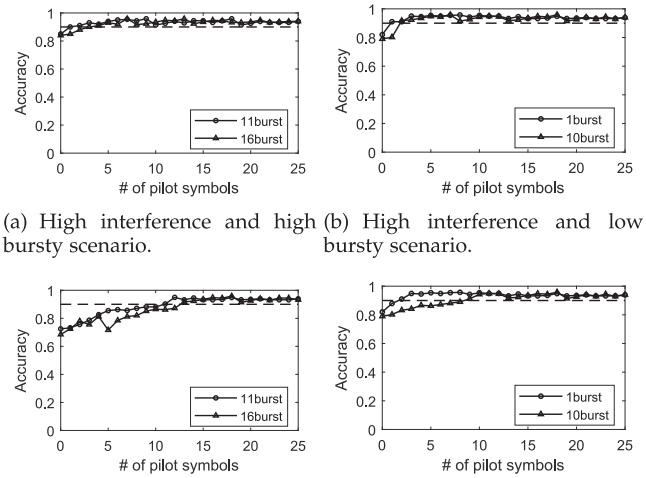
$$\begin{aligned} S = \{s | & RSSI[s-1] - RSSI_{base} < thd, \\ & RSSI[s] - RSSI_{base} \geq thd\} \\ E = \{e | & RSSI[e] - RSSI_{base} \geq thd, \\ & RSSI[e+1] - RSSI_{base} < thd\}. \end{aligned} \tag{13}$$

The bursty level is computed as the average bursty length.

Fig. 6 shows relative errors of estimating PHY-layer features under four different interference scenarios (as shown in Table 1). The label *SINR-xburst* means the feature per-byte SINR is evaluated under bursty level x. Only the results of bursty levels 1, 10, 11, 16 are shown due to space limitation.

Empirical Analysis of the Impact of Interference Patterns on PHY-layer Features. The accuracy of per-packet RSSI and per-packet LQI has been validated to maintain relatively high in most interference patterns [20].Therefore, the four interference patterns have little impact on above two PHY-features.

The accuracy of per-byte SINR is highly affected by the measured noise power that could be interfered. The noise power is recorded when the link turns to idle. When the interference strength is large (i.e., $PAPR > 3.2$), the received power is highly possible to be larger than the CCA threshold, thus we can filter the interfered noise and sample the correct noise power with high probability. Therefore, the relative error of inferred per-byte SINR is small (as shown in Figs. 6a and 6b). When interference strength is small ($1 < PAPR < 3.2$), the interfered noise power could be smaller than the CCA threshold and then is recorded. Figs. 6c and 6d present the relative error of inferring PHY-layer features under LIHB and LILB. First, we see the relative error is increasing with the increased PAPR (i.e., from 1.3 to 2.4), because the WiFi interference starts to appear and leads to interfered PHY-layer features. Then, the relative error is decreasing with the increased PAPR (i.e., from 2.4 to 3.2), because the interference power starts to become larger and the interfered PHY-layer features can be filtered by the CCA threshold. Thus, the



(a) High interference and high bursty scenario.
(b) High interference and low bursty scenario.
(c) Low interference and high bursty scenario.
(d) Low interference and low bursty scenario.

Fig. 7. Impact of the number of instrumented pilot symbols on corruption estimation accuracy.

PeakPapr is set to 2.4. Besides, the relative error is increased when bursty level is high because the more the consecutive corrupted bytes are, the higher the probability of recording interfered noise power is.

Fig. 7 shows the results of corruption estimation accuracy when instrumenting different number of pilots. We can find that under high interference power scenarios, due to the limited impact on the accuracy only <2 pilots are enough to achieve $>90\%$ accuracy when the Logistic Regression model is converged. Fig. 6 guide us to set the upper bound of the needed pilots (i.e., MaxPilot = 13, and MinPilot = 2).

Algorithm 1. Interference Pattern Calculation at Receiver

Input : RSSI sequence RSSI[L]
Output : Interference pattern IPT
PAPR = get_papr(RSSI[L]);
bursty = get_bursty_level(RSSI[L]);
burst_p = 0; PAPR_d = 0;
if $PAPR > HighInterf$ **then**
  return 0;
if $bursty \leq MaxBurst$ **then**
  burst_p = (bursty - MinBurst)/(MaxBurst - MinBurst);
else
  burst_p = 1;
if $PAPR \leq PeakPapr$ **then**
  PAPR_d = (PAPR - MinPapr)/(PeakPapr - MinPapr);
else
  PAPR_d = 1 - (PAPR - PeakPapr)/(MaxPapr - PeakPapr);
IPT = PAPR_d*burst_p;
return IPT;

Calculating Interference Pattern. Based on the above observations, we design a simple but effective interference pattern calculation algorithm to guide the instrumentation of pilots (outlined in Algorithm 1). The input of the algorithm is the in-packet RSSI time series sampled during packet reception. The output of the algorithm is the interference pattern which determines the number of instrumented pilots. We first detect different interference power level according to HighInterf, then the interference pattern is computed by the multiplication of PAPR_d (i.e., the relative ratio to the PeakPapr) and burst_p (i.e., the ratio to the range of bursty level) as shown in line 14 of Algorithm 1.

Adaptive Pilot Instrumentation at Sender. At the sender side, when there is a packet arrived from the network layer, AccuEst first finds the feedback information in the pilot_buffer corresponding to this packet id. If AccuEst can find the corresponding feedback and the feedback is non-zero, AccuEst would instrument extra sup_pilot pilots into the packets. To let the pilot capture more information at the receiver side, the instrumented packets are interleaved before sending out as shown in Algorithm 2.

Algorithm 2. Adaptive Pilot Instrumentation at Sender

case *pkt received from network layer* **do**
  find sup_pilot in pilot_buffer corresponding to this pkt_id;
  if *sup_pilot is not null* **then**
    instrument sup_pilot pilots to the end of pkt header;
  interleaving the pkt under symbol level;
  send pkt and start ACK timer;
case *ACK timer times out* **do**
  notify upper-layer protocol ACK timer times out;
case *ACK/NACK is received* **do**
  extract (sup_pilot, pkt_id) from ACK/NACK;
  store (sup_pilot, pkt_id) into pilot_buffer;
  notify upper-layer protocol ACK/NACK;

Adaptive Pilot Instrumentation at Receiver. When a packet arrives and is de-interleaved, the receiver needs to complete four tasks as shown in Algorithm 4. (1) Obtaining pilots. The receiver first detects the biased bits and then classifies the biased type of symbols. AccuEst determines whether the symbol is treated as a link layer information following the Algorithm 3. AccuEst directly regards the biased symbols as the link layer information. For the intermediate symbols, only when the intermediate symbols are determined as corrupted AccuEst can then treat them as the link layer information. AccuEst determines the symbol as corrupted as long as one of the biased bits in the symbol are different from the received bits. We then append extra pilots into link layer information according to the corresponding sup_pilot found in pilot_buffer.

(2) Discarding abnormal pilots. When the link is unreliable, the feedback information could be lost at the sender, resulting in that the receiver extracts wrong pilots. When the difference of corruption estimation results inferred by pilots and our model exceeds a threshold AbPilots, the receiver regards the pilots as abnormal and discards them. (3) Updating the model and detecting corruptions. The model is updated when there are available pilots. Besides detecting corruptions, the receiver can calculate symbol error rate from estimated corruptions. (4) Calculate the extra needed pilots for the next transmission. It is sent back to the sender using ACK/NACK. The key-value pair (sup_pilot, n_packet_id) are stored into pilot_buffer.

5 Evaluation

In this section, we evaluate the performance of AccuEst, and compare AccuEst with the state-of-the-art, i.e., CARE [16].

Fig. 8. 8x10 indoor testbed.

We also deploy AccuEst on an indoor testbed running CTP protocol [36] to further evaluate the system and discover its benefits to assist a coding-based transmission protocol (i.e., ACR [37]). The indoor testbed consists of 8x10 TelosB nodes and 30 of them are used in our experiments (see Fig. 8). AccuEst is implemented on the TelosB platform running TinyOS 2.1.1. The code size is ∼9.1 KB in ROM, and ∼3.8 KB in RAM. Considering a TelosB node has a total of 48 KB ROM and 10 KB RAM, this overhead is acceptable.

Algorithm 3. Link Layer Information Extraction at Receiver

```
Input: Biased symbol dictionary sym_biased{}, packet
       header pkt_header[]
Output: Link layer information link[], corresponding
        symbols position pos_link[]
for every sym_pos do
  ideal_symbol = sym_biased[sym_pos];
  rec_symbol = pkt_header[sym_pos];
  get ideal_symbol type and store into sym_type;
  case sym_type is BIASED do
    if ideal_symbol equals rec_symbol then
      append 0 to link[];
    else
      append 1 to link[];
    append sym_pos to pos_link[];
  case sym_type is INTERMEDIA do
    if ideal_symbol not equals rec_symbol then
      append 1 to link[];
      append sym_pos to pos_link[];
return pos_link[] and link[];
```

5.1 Experimental Methodology

We use WiFi AP with 802.11g to generate interference. Setting to 54 Mbps and 2,000 bytes packet length, which is the common settings. We select overlapped channels for WiFi and CC2420 radio (i.e., channel 11 for WiFi and channel 21 for CC2420 radio). To present different WiFi traffic patterns [7], [20] (i.e., web browsing and video streaming), we use iperf [38] with 802.11g WiFi AP to generate different bursty levels (i.e., 3∼6M, 9∼12M TCP traffic, and 15∼18M UDP traffic, respectively). The WiFi packet length is set to 2,000 bytes with 54 Mbps data rate. To present different noise environments (i.e., power control room and transformer vault [30]), we use USRP [34] to generate background noise levels (i.e., −85∼−82 dBm, −90 dBm and −98∼−95 dBm). To present different SINR, we vary the distance between the transceiver pair and the interferer for different noise environments to generate SINR ranges (e.g., −10∼−6 dB, −8∼−3 dB, 0∼4 dB). With the above tools, we can simulate practical interference scenarios like office or noisy industry environments [1], [7], [30].

TABLE 2
Experimental Settings

Scenarios	SINR	Traffic	Noise power
HIHB (Nearby video streaming, office)	−5∼−1 dB	UDP 15∼18 M	−90 dBm
HILB (Nearby browser, office)	−5∼−1 dB	TCP 3∼6 M	−90 dBm
LIHB (Faraway video streaming, office)	0∼ 4 dB	UDP 15∼18 M	−90 dBm
LILB (Faraway browser, office)	0∼ 4 dB	TCP 3∼6 M	−90 dBm
HIHN (Nearby interf., power control room)	−10∼ −6 dB	TCP 9∼12 M	−85 ∼ −82 dBm
HILN (Nearby interf., transformer vault)	−8∼−3 dB	TCP 9∼12 M	−98 ∼ −95 dBm
LIHN (Faraway interf., power control room)	−3∼ 2 dB	TCP 9∼12 M	−85 ∼ −82 dBm
LILN (Faraway interf., transformer vault)	1∼ 6 dB	TCP 9∼12 M	−98 ∼ −95 dBm

H(high), L(low), I(interference), B(bursty), N(noise).

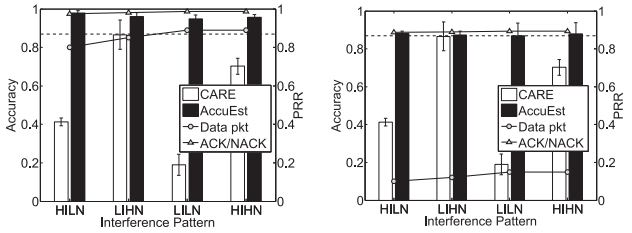
Single-hop Experiment Settings. We use two TelosB nodes running TinyOS 2.1.1 as a transceiver pair. They communicate with each other using the CC2420 radio with the power level of 6 at a distance of 1.5m apart. The sender sends data packets with a payload of 97 bytes to the receiver with an interval of 512 ms.

Multi-hop Testbed Experiment Settings. As shown in Fig. 8, the distance between any two nodes is 0.5 m. We set the radio power of each node to -32.5 dBm, resulting in a multi-hop wireless sensor network. We apply AccuEst to a coding-based transmission protocol ACR [37] to further evaluate the system. ACR relies on detected corruptions to determine the retransmitted partial packet when the decoding procedure fails. We replace the corruption detection component in ACR and CARE with our approach, respectively. We then compare the end-to-end performance metrics, i.e., packet delivery rate and data latency. The detailed experimental settings are shown in Table 2.

Three practical scenarios (i.e., web surfing, file download and mixed) are set up to evaluate AccuEst in terms of the throughput and the data yield. The throughput is the number of bits transmitted from the source node to the sink node per second. The data yield is the ratio between the amount of data packets received at the sink and the total amount of data packets generated by sensor nodes. When continuous failures happen and retransmission rounds exceed the retransmission threshold, the data may be lost. Five laptops are used to surf the Internet via WiFi in the testbed room. The interference types are controlled by performing different actions: web surfing (baidu news), file download (Thunder) and mixed.

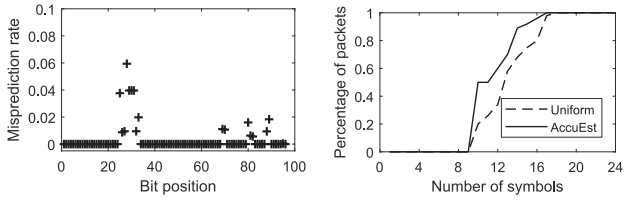
5.2 Corruption Detection Accuracy

As analyzed in Section 3.1, the ΔRSSI indicator suffers from low accuracy in a highly noisy environment (i.e., power



(a) Corruption estimation under reliable link. (b) Corruption estimation under unreliable link.

Fig. 9. Corruption estimation accuracy.



(a) Misprediction rate of extracted biased bits. (b) CDF of the number of extracted pilot symbols.

Fig. 10. Pilot extraction accuracy.

control room and transformer vault [30]). We compare AccuEst with CARE in terms of corruption detection accuracy under different noisy environments and link reliability. Since CARE is a retransmission protocol, we only compare AccuEst with the corruption estimation component of CARE.

Algorithm 4. Adaptive Pilot Instrumentation at Receiver

```
1  pilot extraction window PW[]; sliding window W[];
2  max/min number of needed pilot MaxPilot/MinPilot;
3  PilotR = MaxPilot - MinPilot;
4  case SFD rising do
5     start RSSI sampling and store value in RSSI[L];
6  case SFD falling do
7     stop RSSI sampling;
8  case pkt received do
9     de-interleaving this pkt;
10    if pkt is correct then
11       store the pkt header into PW[];
12    detect biased symbols sym_biased from PW[];
13    find sup_pilot in pilot_buffer with the pkt_id;
14    merge pilot sym_biased and sup_pilot as link[];
15    extract feature phy[] corresponding to link[];
16    if (phy[], link[]) is abnormal then
17       discard link[];
18    else
19       put phy[], link[] into W[] and update model;
20    calculate interference pattern IPT;
21    if IPT is abnormal then
22       IPT = 1;
23    len_symb = len(sym_biased);
24    n_sup_pilot = MinPilot + IPT*PilotR - len_symb;
25    store (n_sup_pilot, n_pkt_id) into pilot_buffer;
26    if pkt is correct then
27       reply ACK (n_sup_pilot, n_pkt_id);
28    else if pkt is error then
29       calculate feature phy[L] from RSSI[L];
30       estimate corruption cor_pos[] from phy[L];
31       calculate SER according to cor_pos[];
32       deliver_to_upperprotocol(cor_pos[], SER, pkt);
33       reply NACK (n_sup_pilot, n_pkt_id);
```

As shown in Fig. 9a, AccuEst consistently achieves higher accuracy than CARE under all scenarios. Specifically, AccuEst improves accuracy significantly by 79.4 percent on average compared to CARE. The reason is that when SINR is extremely low (e.g., $-10\sim-6$ dB) or higher than -3 dB, and noise level is comparable to or higher than interference strength (i.e., HIHN, LIHN and LILN), CARE would misjudge the byte as correct since the RSSI difference is negligible in such cases. However, AccuEst will classify the byte as erroneous since large noise makes SINR small. Moreover, AccuEst can adapt to various RSSI sensitivity of nodes using Logistic Regression model, resulting in a higher accuracy than CARE when the RSSI difference is large (i.e., HILN).

To evaluate the impact of the unreliable link on AccuEst, we reduce the power level of two TelosB nodes to 2, at a distance of 1.5 m. With this configuration, the received signal strength is low, resulting in an unreliable link ($<15\%$ PRR for data packet). Fig. 9b presents the corruption detection accuracy for the unreliable link. Comparing Figs. 9a and 9b, we see that the accuracy reduction of AccuEst is small. The reason is two-fold. First, the lost of feedback information (i.e., interference pattern and packet id) may reduce the accuracy of AccuEst. However, the size of ACK/NACK packet including the feed information is much smaller than typical data packet. Therefore, as shown in Fig. 9b, the loss rate of ACK/NACK packet (about 10.3 percent) is much smaller than that of the data packet (about 75.7 percent). This mitigates the impact of unreliable link. Second, when the feedback information is lost, AccuEst incorporates an effective method to discard abnormal pilots. Therefore, the pilots instrumented at unsynchronized positions between the sender and the receiver are abandoned. This further improves the robustness of AccuEst against unreliable link.

5.3 Pilot Extraction Performance

We evaluate the performance of the pilot extraction algorithm. We update the pilot observation window size every 10 seconds. We repeat experiments 10 times under different scenarios as detailed in Section 5.1, and the results are averaged. Fig. 10a shows the misprediction rates for the bits located in the first 12 bytes. The results show that the estimated pilot observation window size can effectively bound the misprediction rate. Fig. 10b presents the CDF of the extracted pilot symbols. The intermediate pilot symbols are included only when they are determined as corrupted. The baseline algorithm Uniform utilizes a fixed observation window size to detect biased bits. Results show that the average number of pilot symbols is 10 and AccuEst outperforms the Uniform algorithm by 9.3 percent, which validates the effectiveness of the pilot symbol extraction algorithm in terms of training the corruption estimation model.

5.4 Computation Overhead

The major computation overhead of AccuEst falls into the number of pilots that are used to update the parameters. The number of pilots is adjusted along with the change of interference pattern. Therefore, we evaluate computation overhead under different interference patterns. The results are obtained from 1,000 packets under four interference

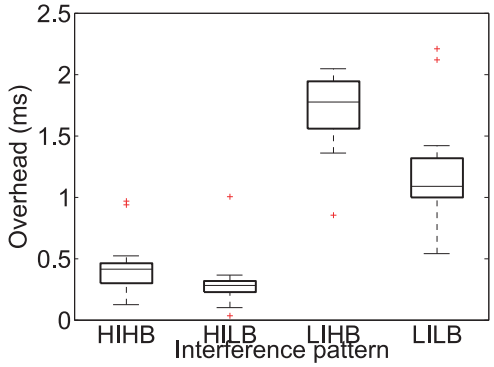


Fig. 11. Computation overhead.

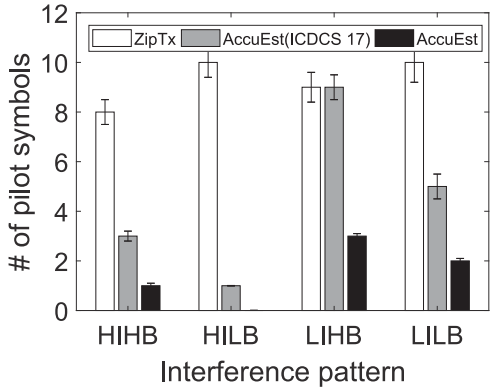


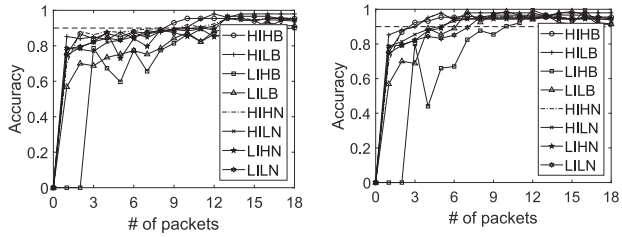
Fig. 12. Pilot overhead.

patterns (in Fig. 11). Even in the worst case (LIHB) the median computation overhead is only about 1.803 ms, and it is acceptable considering the MAC backoff time is 5 ms in expectation, and the packet transmission time is 3.5 ms for a 110 bytes packet given a data rate of 250 kbps. In the worst case (LIHB), the computation overhead of the classification is about 9.5 percent of whole computation overhead (e.g., feature calculation, parameter update and corruption classification). It is comparable to feature calculation (about 10.3 percent), but smaller than parameter update (about 90.2 percent). The reason is that parameter update needs more iterations for multiple packets in a sliding window, while both classification and feature calculation only deal with the received packet.

5.5 Pilot Overhead

As described in Section 4, we utilize the number of detected corruptions to estimate the SER in a packet. We thus compare the pilot overhead among AccuEst, AccuEst(ICDCS 17) [31] and the pilot-based approach, i.e., ZipTx [11], with respect to the relative error of estimating SER. The pilot overhead is obtained by averaging the number of instrumented pilots when achieving the same relative error under different interference patterns.

Fig. 12 shows that AccuEst reduces the pilot overhead significantly by 83.7 and 66.7 percent on average compared to ZipTx and AccuEst. The reason is that AccuEst can not only infer SER using the trained Logistic Regression model, but also can extract pilot symbols from existing packet header without incurring too much pilot overhead. While AccuEst still need to instrument extra pilot symbols update the model. As for ZipTx, it always needs to instrument a number of pilots to achieve the same accuracy of SER estimation.



(a) Without symbol level interleaving. (b) With symbol level interleaving.

Fig. 13. Convergence speed.

5.6 Convergence Speed

We now investigate the convergence speed of the Logistic Regression model under different interference scenarios. To evaluate how the corruption detection accuracy evolves, we consider whether there is the symbol-level interleaving component before sending out packets. The experiment is repeated 10 times for each scenarios and the results are averaged. As shown in Fig. 13, we can find that with the symbol-level interleaving, the average number of packets for model convergence is reduced about 51.8 percent compared to that without the symbol-level interleaving.

As shown in Fig. 13, we observe that the worst case is LIHB. The reason is that the LIHB scenario would change the SINR model with high probability. Then we need more pilots to converge the dynamic changing model. Under the LIHB scenario, after the pilot observation window size is determined,only 11 packets (i.e., about 1.035s for 110 bytes packet, given 250 kbps data rate and 0.1s inter-packet interval) are needed to make AccuEst achieve higher than 90 percent accuracy. This number is quite consistent for all 10 packet traces on average.

5.7 Impact of Dynamic Interference Pattern

In this experiment, we evaluate our algorithm's performance under a dynamic interference experiment. The number of pilots is small under high interference scenarios, because the SINR model is relatively stable and AccuEst does not need to frequently update the model. Therefore, we only evaluate the impact of high SINR scenarios (e.g., 0∼4 dB) on the performance of adaptive pilot instrumentation component. We set SINR to range [0, 4] dB while varying bursty levels. The low-bursty level fraction is computed as the ratio of the low bursty time duration and the total duration (i.e., 100 minutes in our evaluation). We compare the pilot overhead among AccuEst, AccuEst(ICDCS 17) and ZipTx when achieving lower than 10 percent relative error of estimated SER.

We conduct each experiment 10 times and show the average results in Fig. 14. The results show that our approach significantly reduces the pilot overhead by 83.1 and 73.8 percent on average compared with ZipTx and AccuEst, while achieving the approximately equivalent relative error compared to ZipTx. The reason is that to achieve a small relative error of estimated SER under any scenarios, ZipTx has to set the number of pilots according to the worst case. Although AccuEst can adapt to the interference patterns to reduce the expectation number of pilots, AccuEst can further reduce the pilot overhead by extracting pilot symbol from existing packet header.

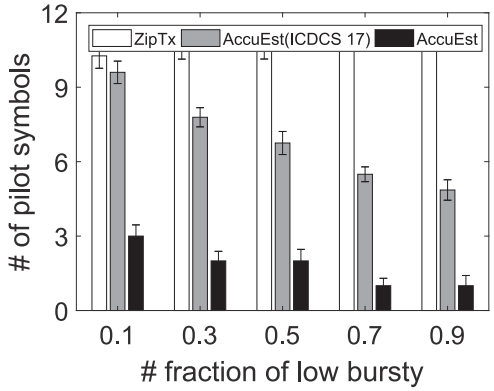
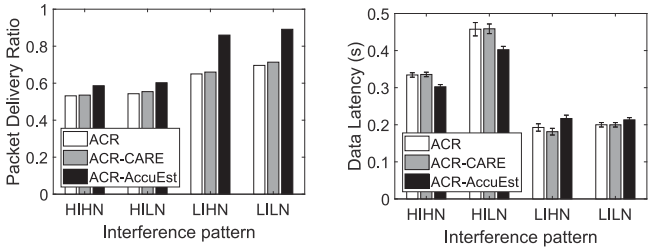


Fig. 14. Impact of dynamic interference patterns.



(a) Packet delivery ratio. (b) Data latency.

Fig. 15. End-to-end performance comparison.

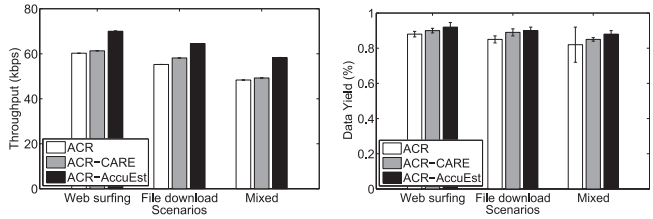
5.8 Testbed Experiments

ACR actively converts most potential collisions into the long and short packet collision patterns to enable a lightweight FEC scheme to recover collided packets. By default, a short ACR packet has 25 bytes data with a 16-bit CRC, and a long ACR packet consists of 11 blocks where the first eight blocks are data and the rest are redundancy. The block size is 10 bytes. When the number of erroneous blocks is larger than three, ACR relies on the detected corruptions to determine the retransmitted partial packets for FEC decoding. If the decoding procedure fails, ACR switches to the ARQ scheme. A block is regarded as corrupted when the difference between $RSSI[i]$ and $RSSI_{base}$ is larger than 5. We replace the corruption detection component with AccuEst and CARE, respectively. A block is regarded as corrupted when there exists at least one erroneous byte.

Packet Delivery Ratio. Fig. 15a shows the packet delivery ratio (PDR) with different corruption detection approaches. The result shows that ACR-AccuEst improves PDR by 22.1 and 19.5 percent on average compared to ACR and ACR-CARE, respectively. The PDR of ACR and ACR-CARE is low under HIHN, LIHN and LILN scenarios. The reason is that both ACR and ACR-CARE suffer from low accuracy of corruption estimation under these scenarios. In HIHN and LIHN, false negatives are more likely to happen (erroneous blocks are identified as correct). The FEC decoding would fail and multiple retransmissions will be incurred. In LILN, false positives are more likely to happen (correct blocks are identified as erroneous), leading to multiple retransmissions of the correct parts of the packets.

Data Latency. The overhead of successfully transmitting a packet at each hop includes three parts: (1) ACR encoding and decoding. (2) MAC backoff and packet transmission. (3) Corruption estimation. The overhead of ACR encoding and decoding is 0.1 and 0.5 ms, respectively, according to our experimental results. The expected MAC backoff time is 5 ms because the maximum initial backoff time in TinyOS is about 10 ms. The packet transmission time is about 3.5 ms for a 110-byte packet with a data rate of 250 kbps [15]. According to Fig. 11, corruption estimation typically costs 1.412 and 0.376 ms on average, under low and high interference scenarios. Therefore, it is acceptable in terms of the MAC backoff time and packet transmission time.

Fig. 15b shows the data latency in a 5-hop network. The results show that compared to ACR-CARE, ACR-AccuEst reduces data latency by 10.2 percent on average under HIHN and LIHN. The reason is that under HIHN and LIHN, the symbol error rate is relatively high, the corrupted blocks that are missed by ACR and ACR-CARE would lead to multiple FEC decoding failures and retransmissions of whole packet, resulting in large latency. Differently, ACR-AccuEst can detect more corrupted blocks and reduce extra FEC decoding overhead, only requesting the retransmissions of corrupted part of the packets.



(a) Throughput. (b) Data yield.

Fig. 16. Performance evaluation under practical scenarios.

Throughput. Fig. 16a shows that ACR-AccuEst improves the throughput of ACR and ACR-CARE by 17.8 and 14.4 percent on average. The reason is that with more accurate error estimation, ACR-AccuEst improves the efficiency of ACR decoding, resulting in less amount of transmitted bytes for each useful byte. The throughput is thus improved. Note that the throughput improvements under web surfing scenario (e.g., 16.2 and 14.1 percent comparing to ACR and ACR-CARE) is a little bit larger than the one under the file-downloading scenario (e.g., 15.4 and 10.9 percent comparing to ACR and ACR-CARE), because there are more burstiness under the file-downloading scenario and more pilot symbols are instrumented to achieve high error estimation accuracy.

Data Yield. Fig. 16b shows that ACR-AccuEst increases data yield of ACR and ACR-CARE by 5.6 and 2.2 percent on average. The reason is that with accurate error estimation, ACR-AccuEst achieves higher packet reception ratio, resulting in less packet loss. The data yield is therefore increased.

6 Discussion

Energy Efficiency when Enabling High-resolution RSSI Sampling Procedure. The high-resolution RSSI sampling will not incur considerable energy consumption. The reason is two-fold: 1) The RSSI value in the register is averaged over 8 symbol periods (128 μs) according to the CC2420 datasheet [32]. Hence, our implementation does not change hardware sampling and only increases the register reading rate, which incurs small energy consumption compared to radio

operations. For example, TelosB [39] nodes consume 21.8 mA when receiving a packet. The additional RSSI reading only adds 0.5 mA [40]. Note that our approach does not turn CPU into active during duty cycling. It turns the CPU into active when receiving a packet, which lasts for at most 4 ms (considering a max packet length of 128 bytes). 2) Although the energy consumption is slightly increased by RSSI readings, the energy efficiency can be improved since retransmissions can be reduced with our accurate corruption estimation.

Interference Pattern Calculation when Multiple Interference Types are Collided. When different interference types are collided while one of them is dominated, the two features (e.g., bursty length and PAPR) can still effectively reflect the interference patterns according to existing works [17], [20], and our algorithm can thus perform well in this case. When different interference types are collided and none of them is dominated, interference patterns may vary quickly. The number of needed pilot symbols derived from the interference patterns may not timely capture the dynamic interference pattern. i.e., the estimated maximum needed pilot symbols is underestimated or overestimated from the optimal one. In this case, we conservatively set as maximum the number of needed pilot symbols to guarantee the high error estimation accuracy (e.g., as shown in Algorithm 4 line 22-23). With most pilot symbols extracted from packet headers (e.g., about 10 pilot symbols with larger than 95 percent accuracy), fewer extra pilot symbols are needed in AccuEst comparing with AccuEst (ICDCS 17).

Therefore, when multiple interference types are collided, the algorithm of interference pattern calculation may not timely capture the dynamic patterns, but AccuEst can still achieve relatively high overall performance (e.g., high estimation accuracy) by incurring only little overhead (e.g., a small number of instrumented pilot symbols).

Generality on Other Platforms. The core component of our approach is high-resolution RSSI sampling. We believe that for any platform that can enable high-resolution RSSI sampling (e.g., Micaz[41]) or provide fine-grained Channel State Information (CSI) of the channel where the bits are transmitted on (e.g., 5,300 NIC[42]), our approach can be easily applied.

Other Interference Types. We have built the model according to the sampled RSSI, which is adaptive to different types of interference. Our approach performs well when there are clear interference patterns and may not perform well otherwise. We also note that when the interference is random, our approach can still perform better than ZipTx as shown in Fig. 14.

Limitation. Considering fast transmission rate of 802.11n (e.g., typical data rate of 200 Mbit/s [43]), the typical packet on-air time is 5.45 or 57.3 μs according to Esense [44]. Our approach with high-resolution RSSI sampling (e.g., one sample per 32 μs) may not be able to capture the corruption pattern caused by short WiFi packets.

7 Conclusion

Accurate corruption estimation is one of the key factors in improving the resilience of ZigBee transmissions. This paper reveals the limitations of existing in-packet RSSI approaches, and uses per-byte SINR to detect corruption. We combine the link-layer information (pilot symbols) and the PHY-layer features (i.e., per-packet LQI, per-packet RSSI, and per-byte SINR) to further improve corruption detection accuracy. Furthermore, we design an interference pattern-aware pilot instrumentation scheme to strike a good balance between accuracy and overhead. We conduct extensive experiments including single-link and multi-hop to evaluate our approach. Testbed results show that our approach significantly improves packet delivery ratio.

For our future work, there are multiple directions to explore. First, we will generalize our approach to more interference types. Second, we will develop a better instrumentation scheme to further reduce the pilot overhead.

Acknowledgments

This work is supported by the National Science Foundation of China under Grant No. 61772465, Zhejiang Provincial Natural Science Foundation for Distinguished Young Scholars under Grant No. LR19F020001, and the Fundamental Research Funds for the Central Universities (No. 2017FZA5013), the National Natural Science Foundation of China (No. 61602095), the National Postdoctoral Program for Innovative Talents (No. BX201700046), and the Australian Research Council Discovery Grant (No.DP180103932).

References

[1] C. J. M. Liang, N. B. Priyantha, J. Liu, and A. Terzis, "Surviving Wi-Fi interference in low power ZigBee networks," in *Proc. 8th ACM Conf. Embedded Netw. Sensor Syst.*, 2010, pp. 309–322.

[2] X. Guo, M. Mohammad, and M. C. Chan, "Oppcast: Exploiting spatial and channel diversity for robust data collection in urban environments," in *Proc. of 15th ACM/IEEE Int. Conf. Inf. Process. Sensor Netw.*, 2016, pp. 1–12.

[3] K. Jamieson and H. Balakrishnan, "PPR: Partial packet recovery for wireless networks," in *Proc. Conf. Appl. Technol. Archit. Protocols Comput. Commun.*, 2007, pp. 409–420.

[4] S. Katti and D. E. A. Katabi, "Symbol-level network coding for wireless mesh networks," in *Proc. of ACM SIGCOMM Conf. Data Commun.*, 2008, pp. 401–412.

[5] H. Shafagh, J. Gross, S. Duquennoy, A. Hithnawi, and S. Li, "CrossZig: Combating Cross-Technology Interference in Low-power Wireless Networks," in *Proc. 15th ACM/IEEE Int. Conf. Inf. Process. Sensor Netw.*, 2016, pp. 1–12.

[6] R. Koetter and A. Vardy, "Algebraic soft-decision decoding of Reed-Solomon codes," *IEEE Trans. Inf. Theory*, vol. 49, no. 11, pp. 2809–2825, Nov. 2003.

[7] Z. Zhao, W. Dong, G. Chen, G. Min, T. Gu, and J. Bu, "Embracing corruption burstiness: Fast error recovery for ZigBee under Wi-Fi interference," *IEEE Trans. Mobile Comput.*, vol. 16, no. 9, pp. 2518–2530, Sep. 2017.

[8] W. Du, Z. Li, J. C. Liando, and M. Li, "From rateless to distanceless: Enabling sparse sensor network deployment in large areas," in *Proc. 12th ACM Conf. Embedded Netw. Sensor Syst.*, 2014, pp. 312–313.

[9] W. Du, Z. Li, J. C. Liando, and M. L., "From rateless to distanceless: Enabling sparse sensor network deployment in large areas," *IEEE/ACM Trans. Netw.*, vol. 22, no. 4, pp. 2498–2511, Aug. 2016.

[10] S. Sen, N. Santhapuri, R. R. Choudhury, and S. Nelakuditi, "AccuRate: Constellation based rate estimation in wireless networks," in *Proc. 7th USENIX Conf. Netw. Syst. Des. Implementation*, 2010, pp. 12–12.

[11] C. J. Lin, N. Kushman, and D. Katabi, "ZipTx: Harnessing partial packets in 802.11 networks," in *Proc. 14th ACM Int. Conf. Mobile Comput. Netw.*, 2008, pp. 351–362.

[12] J. Huang, Y. Wang, and G. Xing, "LEAD: Leveraging protocol signatures for improving wireless link performance," in *Proc. 11th Annu. Int. Conf. Mobile Syst. Appl. Serv.*, 2013, pp. 333–346.

[13] L. Wang, X. Qi, J. Xiao, K. Wu, M. Hamdi, and Q. Zhang, "Wireless rate adaptation via smart pilot," in *Proc. IEEE 22nd Int. Conf. Netw. Protocols*, 2014, pp. 409–420.

[14] M. Vutukuru, H. Balakrishnan, and K. Jamieson, "Cross-layer wireless bit rate adaptation," in *Proc. ACM SIGCOMM Conf. Data Commun.*, 2009, pp. 3–14.
[15] J.-H. E. A. Hauer, "Mitigating the effects of RF interference through RSSI-based error recovery," in *Proc. Eur. Conf. Wireless Sensor Netw.*, 2010, pp. 224–239.
[16] W. Dong, J. Yu, and X. Liu, "CARE: Corruption-aware retransmission with adaptive coding for the low-power wireless," in *Proc. IEEE IEEE 23rd Int. Conf. Netw. Protocols*, 2015, pp. 235–244.
[17] X. Zheng, Z. Cao, J. Wang, Y. He, and Y. Liu, "ZiSense: Towards interference resilient duty cycling in wireless sensor networks," in *Proc. 12th ACM Conf. Embedded Netw. Sensor Syst.*, 2014, pp. 119–133.
[18] D. Liu, Z. Cao, M. Hou, and Y. Zhang, "Frame counter: Achieving accurate and real-time link estimation in low power wireless sensor networks," in *Proc. 15th ACM/IEEE Int. Conf. Inf. Process. Sensor Netw.*, 2016, pp. 1–12.
[19] A. Hithnawi, H. Shafagh, and S. Duquennoy, "TIIM: Technology-independent interference mitigation for low-power wireless networks," in *Proc. 14th Int. Conf. Inf. Process. Sensor Netw.*, 2015, pp. 1–12.
[20] F. E. A. Hermans, "SoNIC: Classifying interference in 802.15.4 sensor networks," in *Proc. ACM/IEEE Int. Conf. Inf. Process. Sensor Netw.*, 2013, pp. 55–66.
[21] J. Meng, et al., "Smogy-link: Fingerprinting interference for predictable wireless concurrency," in *Proc. IEEE 24th Int. Conf. Netw. Protocols*, 2016, pp. 1–10.
[22] F. M. Sallabi, A. M. Gaouda, A. H. El-Hag, and M. M. A. Salama, "Evaluation of ZigBee wireless sensor networks under high power disturbances," *IEEE Trans. Power Delivery*, vol. 29, no. 1, pp. 13–20, Feb. 2014.
[23] J. M. Cano-Garcia, et al., "An empirical evaluation of the consumption of 802.15.4/ZigBee sensor motes in noisy environments," in *Proc. Int. Conf. Netw. Sens. Control*, 2011, pp. 439–444.
[24] S. S. Hong and S. R. Katti, "DOF: A local wireless information plane," in *Proc. ACM SIGCOMM Conf.*, 2011, pp. 230–241.
[25] K. Jamieson, P. Levis, R. Fonseca, and O. Gnawali, "Four-bit wireless link estimation," in *Proc. 6th Workshop Hot Topics Netw.*, 2007, pp. 1–7.
[26] D. S. J. De Couto, D. Aguayo, J. Bicket, and R. Morris, "A high-throughput path metric for multi-hop wireless routing," in *Proc. 9th Annu. Int. Conf. Mobile Comput. Netw.*, 2003, pp. 134–146.
[27] S. Min Kim, et al., "Exploiting causes and effects of wireless link correlation for better performance," in *Proc. IEEE Conf. Comput. Commun.*, 2015, pp. 379–387.
[28] S. R. Das, R. Maheshwari, and S. Jain, "On estimating joint interference for concurrent packet transmissions in low power wireless networks," in *Proc. 3rd ACM Int. Workshop Wireless Netw. Testbeds Experimental Eval. Characterization*, 2008, pp. 89–94 .
[29] T. Liu and A. E. Cerpa, "TALENT: Temporal adaptive link estimator with no training," in *Proc. ACM 10th ACM Conf. Embedded Netw. Sensor Syst.*, 2012, pp. 253–266.
[30] V. C. Gungor, B. Lu, and G. P. Hancke, "Opportunities and challenges of wireless sensor networks in smart grid," *IEEE Trans. Ind. Electron.*, vol. 57, no. 10, pp. 3557–3564, Oct. 2010.
[31] G. Chen, W. Dong, Z. Zhao, and T. Gu, "Towards accurate corruption estimation in ZigBee under cross-technology interference," in *Proc. IEEE 37th Int. Conf. Distrib. Comput. Syst.*, 2017, pp. 425–435.
[32] CC2420, 2007. [Online]. Available: http://focus.ti.com/lit/ds/symlink/cc2420.pdf.
[33] T. Liu and A. E. Cerpa, "Foresee (4C): Wireless link prediction using link features," in *Proc. 10th ACM/IEEE Int. Conf. Inf. Process. Sensor Netw.*, 2011, pp. 294–305.
[34] E. R. LLC, 2007. [Online]. Available: https://www.ettus.com/.
[35] M. Spuhler, V. Lenders, and D. Giustiniano, "BLITZ: Wireless link quality estimation in the dark," in *Proc. Eur. Conf. Wireless Sensor Netw.*, 2013, pp. 99–114.
[36] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis, "Collection tree protocol," in *Proc. 7th ACM Conf. Embedded Netw. Sensor Syst.*, 2009, pp. 1–14.
[37] Y. Wu, G. Zhou, and J. A. Stankovic, "ACR: Active collision recovery in dense wireless sensor networks," in *Proc. IEEE INFOCOM*, 2010, pp. 1–9.
[38] Iperf, 2005. [Online]. Available: http://dast.nlanr.net/projects.
[39] J. Polastre, R. Szewczyk, and D. Culler, "Telos: Enabling ultra-low power wireless research," in *Proc. 4th Int. Symp. Inf. Process. Sensor Netw.*, 2005, pp. 364–369.
[40] R. Fonseca, et al., "Quanto: Tracking energy in networked embedded systems," in *Proc. 8th USENIX Conf. Operating Syst. Des. Implementation*, 2008, pp. 323–338.
[41] J. Hill and D. Culler, "Mica: A wireless platform for deeply embedded networks," *IEEE Micro*, vol. 22, no. 6, pp. 12–24, Nov./Dec. 2002.
[42] D. Halperin, W. Hu, A. Sheth, and D. Wetherall, "Tool release: Gathering 802.11n traces with channel state information," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 1, 2011, Art. no. 53.
[43] IEEE 802.11n, 2009. [Online]. Available: http://luci.subsignal.org/jow/802.11n-2009.pdf.
[44] K. Chebrolu and A. Dhekne, "Esense: Communication through energy sensing," in *Proc. 15th Annu. Int. Conf. Mobile Comput. Netw.*, 2009, pp. 85–96.

Gonglong Chen received the BS degree from the College of Criminal Justice, East China University of Political Science and Law. He is currently working toward the PhD degree in the College of Computer Science, Zhejiang University. His current research interests include wireless and mobile computing. He is a student member of the IEEE.

Wei Dong received the BS and PhD degrees from the College of Computer Science at Zhejiang University in 2005 and 2011, respectively. He is currently a full professor in the College of Computer Science at Zhejiang University. He leads the Embedded and Networked Systems (EmNets) lab in Zhejiang University. He has published over 100 papers in prestigious conferences and journals including MobiCom, INFOCOM, ICNP, *ToN*, *TMC*, etc. His research interests include Internet of Things and sensor networks, wireless and mobile computing, and network measurement. He is a member of the IEEE and the ACM.

Zhiwei Zhao received the PhD degree in computer science from Zhejiang University, in 2015. He is currently an associate professor with the College of Computer Science and Engineering, University of Electronic Science and Technology of China. His research interests include on wireless computing, heterogeneous wireless networks, protocol design, and network coding. He is a member of the IEEE.

Tao Gu received the bachelor's degree from the Huazhong University of Science and Technology, the MSc degree from Nanyang Technological University, Singapore, and the PhD degree in computer science from the National University of Singapore. He is currently an associate professor of computer science with RMIT University, Australia. His current research interests include mobile computing, ubiquitous/pervasive computing, wireless sensor networks, distributed network systems, sensor data analytics, Internet of Things, and online social networks. He is a senior member of the IEEE and a member of the ACM.