On Improving Wireless Channel Utilization: A Collision Tolerance-Based Approach

Xiaoyu Ji, *Member, IEEE*, Yuan He, *Member, IEEE*, Jiliang Wang, *Member, IEEE*, Kaishun Wu, Daibo Liu, *Member, IEEE*, Ke Yi, *Senior Member, IEEE*, and Yunhao Liu, *Fellow, IEEE*

Abstract—Packet corruption caused by collision is a critical problem that hurts the performance of wireless networks. Conventional medium access control (MAC) protocols resort to collision avoidance to maintain acceptable efficiency of channel utilization. According to our investigation and observation, however, collision avoidance comes at the cost of miscellaneous overhead, which oppositely hurts channel utilization, not to mention the poor resiliency and performance of those protocols in face of dense networks or intensive traffic. Discovering the ability to tolerate collisions at the physical layer implementations of wireless networks, we in this paper propose *Coco*, a protocol that advocates simultaneous accesses from multiple senders to a shared channel, i.e., optimistically allowing collisions instead of simply avoiding them. With a simple but effective design, *Coco* addresses the key challenges in achieving collision tolerance, such as precise sender alignment and the control of transmission concurrency. We implement *Coco* in 802.15.4 networks and evaluate its performance through extensive experiments with 21 TelosB nodes. The results demonstrate that *Coco* is light-weight and enhances channel utilization by at least 20 percent in general cases, compared with state-of-the-arts protocols.

Index Terms—Channel utilization, collision tolerance, capture effect, wireless networks, throughput

1 Introduction

Wireless networks suffer from collisions. This is essentially due to the broadcast nature of wireless communications. Simultaneous transmissions in a common channel are likely to interfere with each other and thus cause corruption[1] of the transmitted packets [1], [2], [3], [4]. Without appropriate handling of collisions, network performance like channel utilization is degraded [5]. Hence how to resolve collisions is a crucial issue in the area of wireless networks.

In order to improve wireless channel utilization, an intuitive approach is to avoid collisions. Namely, no more than one sender should transmit concurrently in any specific time slot. Based on the philosophy of *collision avoidance*, many protocols have been proposed in the past decades. Their common principle is to scatter the transmissions along the temporal dimension to limit the chance of collisions. The ability of collision avoidance, however, usually comes at the cost of sacrificing the efficiency of channel utilization [6], [7]. Specifically, TDMA-like protocols incur non-negligible overhead in coordination and synchronization to make schedules. For CSMA protocols, senders always conservatively choose the size of backoff window, e.g., using the Binary Exponential Backoff algorithm, because the potential contention is fundamentally uncertain and difficult to be accurately predicted. As a result, a lot of *idle slots* (i.e., time without packet transmission) are left unused in the channel. That problem becomes even more serious in the scenarios with high node density and intensive traffic load [8].

Based on the above fact, we find there is an inherent conflict between collision avoidance and channel utilization. That motivates us to reconsider the way to handle collisions from a new respect. For the purpose of better utilization of the channel, one can choose to tolerate or even allow collisions instead of simply avoiding them. Following that idea, we make some preliminary attempts on modifying the medium access control mechanism. Fig. 1 plots the comparison between collision avoidance and collision tolerance. Using collision avoidance, nodes take random backoffs against collisions such that packets are separated with numerous idle slots. In comparison, supported by the principle of collision tolerance, the senders can adopt a relatively aggressive strategy in packet transmissions. More than one packet transmissions are allowed to appear in the shared channel at one time (the reason for aligning packets is explained in the next section). The surprising result is that five more packets are successfully transmitted using collision tolerance. The utilization is enhanced by 71.4 percent.

The feasibility of collision tolerance actually comes from the physical layer implementations of wireless networks, e.g., IEEE 802.15.4 [9] and 802.11 networks [10]. For example, the DSSS (direct-sequence spread spectrum) modulation scheme in those implementations increases the

1. We differ collision from corruption in this paper. Collision only indicates the overlap of signals in time domain while corruption means failure in decoding any signal involved in the collision.

• *X. Ji and K. Yi are with Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong. E-mail: jixiaoyu.hk@gmail.com, yike@cse.ust.hk.*
• *Y. He, J. Wang, and Y. Liu are with School of Software and TNLIST, Tsinghua University, China. E-mail: {he, yunhao}@greenorbs.com, aliangwjl@gmail.com.*
• *K. Wu is with the School of Computer Science and Engineering, Shenzhen University, Shenzhen, China. E-mail: kwinson@cse.ust.hk.*
• *D. Liu is with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Sichuan 610051, China. E-mail: dbliu.sky@gmail.com.*

Manuscript received 6 Apr. 2015; revised 13 Dec. 2015; accepted 26 Apr. 2016. Date of publication 12 May 2016; date of current version 2 Feb. 2017.
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.2016.2567380

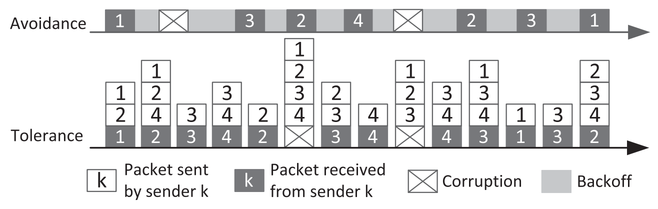


Fig. 1. Channel utilization under collision avoidance and collision tolerance. The ACK packets are omitted for clear illustration.

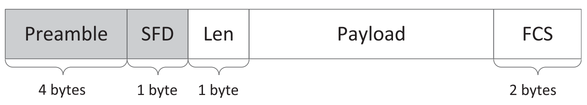


Fig. 2. Format of 802.15.4 packet. In the reception process, preamble is used to detect the beginning of a packet and synchronize the receiver and the sender. After that, SFD (start of frame delimiter) triggers the reception of the packet. Len indicates the length of the payload and FCS means the frame check sequence.

resistance to interference by introducing abundant redundancy. Though collision tolerance demonstrates great potential to improve channel utilization, several critical challenges by far restrict the application of collision tolerance in practice. First, collision tolerance poses stringent timing requirement on transmissions. Concurrent packet transmissions must be aligned. Second, the concurrency of transmissions must be limited. If such issues cannot be well solved, collisions still result in packet corruptions. The details of these two challenges are presented in the next section.

In order to address the above challenges, in this paper we propose *Coco*, a protocol that leverages the ability of collision tolerance to improve channel utilization. Using *Coco*, concurrent senders optimistically contend with each other to access the shared medium. In *Coco*, Senders are triggered by the receiver's notification in order to achieve sender packets alignment. A feedback control mechanism is devised to assign an appropriate transmission probability p and all senders transmit with the probability p. In this way, *Coco* delicately limits the concurrency and achieves near-optimal channel utilization. The contributions of this paper are summarized as follows:

- We propose to tolerate collisions instead of avoiding them. We present the principle behind collision tolerance and theoretically analyze the performance gain brought by collision tolerance, compared with collision avoidance.
- We investigate the challenges in achieving collision tolerance and design the *Coco* protocol to tackle those challenges in practice. *Coco* mainly consists of two parts: the sender alignment mechanism and the feedback control algorithm to adaptively set the transmission probability p for colliding senders.
- We implement *Coco* in 802.15.4 networks and evaluate its performance under a wide variety of network settings. The results show that *Coco* is light-weight and enhances channel utilization by at least 20 percent in general cases, compared with state-of-the-arts protocols.

The rest of this paper is organized as follows. Section 2 presents in detail the challenges to achieve collision tolerance. In Section 3, we present the theoretical model of channel utilization with collision tolerance and analyze the achievable performance gain, compared with collision avoidance. Section 4 introduces the design details of *Coco*, followed by the performance evaluation results in Section 5. We discuss the related work in Section 6 and conclude this paper in Section 7.

2 Challenges in Collision Tolerance

Recent studies on protocol design pay increasing attention to physical layer (PHY) characteristics, which demonstrate potential opportunities in improving network performance [11]. One of the most promising directions is collision tolerance. In IEEE 802.15.4 networks using the DSSS modulation scheme, for example, the tolerance to collisions is manifested as capture effect [2], [12], [13]. Capture effect is a phenomenon that the receiver can correctly receive a stronger signal in face of other signals' interference [14]. This phenomenon enables successful transmission of a packet along with collisions. Collision tolerance, however, is a nontrivial task and induces several challenges in practice. In this section, we investigate the challenges that hinder the exploitation of collision tolerance. We first conduct a series of experiments to look into the behavior of collision tolerance, and then we show through analysis the solutions to overcome those challenges.

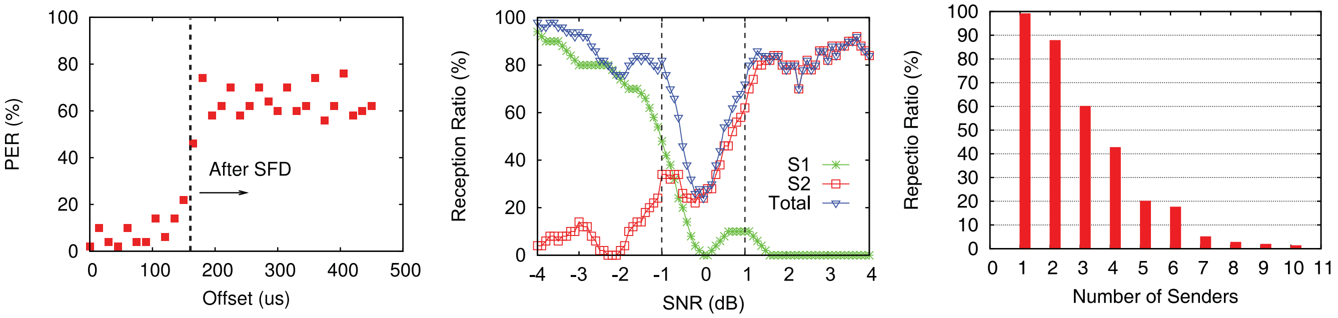
Experiment settings. We conduct our experiments in ContikiOS [15] with TelosB nodes [16], which are typical IEEE 802.15.4 software and hardware platforms. For senders, the carrier sense function is disabled to deliberately generate collisions. In each round of transmission, the receiver first broadcasts a probe packet. On receiving the probe, the senders are triggered to send their packets and therefore collisions occur. The length of the sent packets is 120 bytes. The format of a 802.15.4 packet is shown in Fig. 2.

2.1 Challenge 1: Timing Requirement

We first examine the impact of timing, namely the offset of arrival time among the packets. As described later, the synchronization of arriving packets affects the packet reception result.

Experiment settings. We use three nodes, one is a receiver and the other two are senders (namely A and B). The transmitting power of the senders is set to -5 dBm to ensure high SNR against ambient noise. Sender A is put around 10 feet from the receiver and transmits a packet once it receives the probe packet from the receiver. While for sender B, it waits for a pre-configured offset time Δt before transmitting its packet after receiving the probe. We vary Δt and see the resulting PER (packet error rate) at the receiver side. We place sender B with different distances from the receiver, e.g., from 1 to 20 feet to generate the cases: (1) A is stronger than B (when B is more than 10 feet away from the receiver) and (2) B is stronger than A (when B is less than 10 feet away from the receiver).

Results analysis. As we can see from Fig. 3a, a clear ascent of PER appears when the offset Δt exceeds 160 μs, which is exactly the time duration of the preamble plus SFD (start of frame delimiter). The reason behind is as follows. When the signals overlap with each other, the receiver always tries to find the strongest one by synchronizing to its preamble. Therefore when the offset is less than 160 μs, the strong signal is always captured no matter it comes early or late



(a) Packet error rate vs. packets offset (b) Reception ratio vs. SNR (c) Reception ratio vs. sender number

Fig. 3. Investigation of challenges in exploiting collision tolerance. In (a), the packet error rate has a sharp increase when the offset of arriving packets exceeds a value of 160 μs. (b) shows the reception ratio for the 2-sender case. Even the reception ratio for individual senders (S1/S2) decreases/increases when the relative SNR decreases/increases, the total reception ratio keeps high and degrades only when the two senders have a nearly strong signal level. (c) depicts the reception ratio with a different number of senders, and the reception ratio decreases with the increase of the number of senders.

(even when the preamble of the strong signal interferes with the payload of the weak one, the receiver still performs preamble synchronization operation instead of receiving the payload of the weak one). On the contrary, when the strong signal comes late until after the SFD of the early-arriving weak signal, it acts as strong interference to the weak ones, as the receiver begins receiving the payload of the weak one and is not able to synchronize to the preamble of the strong one. In this case, the weak signal is corrupted by the strong signal, as the failure cases after SFD show in Fig. 3a. Also, the receiver is unable to receive the strong signal in those cases because it misses the SFD of the strong signal. In words, tolerance of collisions requires the offset among the colliding signals be restricted in a certain range, i.e., the time length of the preamble plus SFD. Only in this condition, the receiver is able to identify the strong signal from collisions and receive it.

2.2 Challenge 2: Concurrency Requirement

For the purpose of collision tolerance, the concurrency (the number of concurrent transmissions) must be limited. Otherwise the relatively strong one is not dominating and it is corrupted by the mixed signals of the weak ones. We demonstrate via experiments the opportunity and existing problems with different concurrency.

Experiment settings. We start with an experiment with two senders ($S1$ and $S2$) and one receiver. The experiment scenario is like the above timing experiment and the above-mentioned timing requirement is satisfied. $S1$ is placed stationarily 2 feet away from the receiver while $S2$ is moved towards the receiver from a point 3 feet away from the receiver. At the beginning, $S1$ has a stronger relative SNR than $S2$ (SNR here is calculated by the RSSI difference). When $S2$ approaches the receiver, the SNR of $S1$ decreases while the SNR of $S2$ increases. At some point, they have an equivalent SNR. After that, $S2$ has stronger SNR than $S1$. We record the packet reception ratio (PRR) for $S1$, $S2$ and the total ratio ($S1 + S2$) at the receiver side and plot them to the relative SNR of $S1$ and $S2$.

Results analysis. In Fig. 3b, we can see that with the increase of $S2$'s SNR, the reception ratio of the two senders' packets demonstrates opposite trends. The aggregated reception ratio is high, however. The key finding here is that the aggregated reception ratio degrades only when the two senders have comparably strong power, i.e., a relative SNR near 0 dB in Fig. 3b. That is, for the 2-sender cases, there is extremely large opportunity to exploit collision tolerance and corruption occurs with small probability. In the following we extend the sender number to general cases.

We increase the number of senders from 2 to 10 to further observe collision tolerance. The transmitting power is still -5 dBm and senders are randomly placed around the receiver in a circle with a radius of 30 feet, in order to generate cases with strong and weak signals. For each quantity of senders, we change their positions and repeat the experiments for 100 rounds. Fig. 3c shows that the average reception ratio decreases accordingly when the total number of senders increases. The average reception ratio is nearly zero when there are more than 7 senders. Moreover, the average reception ratio is surprisingly high when the number of senders does not exceed four in our experiment. The result when there are two senders coincides with the observation in Fig. 3b. The experimental results apparently indicate that there is great potential when concurrency is low and we should restrict it to better exploit collision tolerance. Compared with other techniques to achieve collision tolerance, e.g., power control, controlling the transmission concurrency is relatively effective and easy to implement in practice and therefore we adopt it in our protocol.

As a brief summary, collision tolerance is an attractive ability of wireless channels, which poses two critical challenges: the timing requirement and the concurrency requirement. In the subsequent sections, we first theoretically formulate the channel utilization problem with collision tolerance and analyze the corresponding performance in comparison with collision avoidance based approaches. Based on such theoretical foundations, we address the aforementioned challenges in the design and implementation of *Coco* protocol, which will be presented in Section 4.

3 Theoretical Formulation

3.1 Channel Utilization Model

In this part, we model channel utilization as a function of transmission probability p and formulate it into an optimization problem. For convenience, we assume a time slot is the basis element of time. The duration of a packet transmission generally consists of multiple time slots.

At the receiver side, a time slot has in total three possible states as follows:

- *Idle slot*, where there is no transmission to the receiver.
- *Successful slot*, where there is *at least* one sender transmitting and a packet is correctly received.
- *Corrupted slot*, in which multiple senders transmit concurrently and all packets are corrupted.

For each of the above three slot states, we use P_i, P_s and P_c to respectively denote the probability for the state to appear. Specifically, the probability for a slot to be idle is:

$$P_i = (1-p)^N, \tag{1}$$

where N is the total number of senders. Note that in *Coco*, P_s is different from that in traditional protocols. Conventionally, a slot is a successful slot when and only when there is one sender successfully transmitting while others stay silent. With collision tolerance, P_s is increased by exploiting the opportunity of concurrency with 2, 3, 4 and so on. Thus P_s is calculated by:

$$P_s = \sum_{k=1}^{N} \binom{N}{k} p^k (1-p)^{N-k} C(k), \tag{2}$$

where $C(k)$ is the *capture probability* [17], which means the probability that a packet can be correctly received from a collision when k senders are transmitting concurrently. In [17], [18], various models are proposed to measure the capture probability considering the power difference, fading and multi-path effect. We follow the model in [18] which is widely adopted:

$$C(k) = \frac{k}{\sqrt{2\pi}\sigma_s} \int_{-\infty}^{\infty} \left(\int_0^{\infty} [g(\xi, r)]^{k-1} f(r) dr \right) e^{\frac{-\xi^2}{2\sigma_s^2}} d\xi, \tag{3}$$

where

$$g(\xi, r) = \frac{1}{\sqrt{2\pi}\sigma_s} \int_{-\infty}^{\infty} \left(\int_0^{\infty} \frac{f(r_1) dr_1}{1 + z e^{\xi_1 - \xi} (\frac{r}{r_1})^{\beta}} \right) e^{\frac{-\xi_1^2}{2\sigma_s^2}} d\xi_1, \tag{4}$$

in Eqs. (3) and (4), ξ is a Gaussian variable with zero mean and σ^2 variance. r is the distance between the pair of sender and receiver. Eqs. (3) demonstrates the specific capture ratio with the increase of k.

For a corrupted slot, the probability P_c can be calculated by:

$$P_c = 1 - P_i - P_s. \tag{5}$$

To measure channel utilization and derive the performance gain compared with CSMA backoff-based protocols, we model channel utilization as a function of P_i, P_s and P_c (Eqs. (1), (2) and (5)). For a specific time instant, channel utilization is the ratio of a successful slot's duration to the total time:

$$Util(p) = \frac{P_s T_s}{P_s T_s + P_c T_c + P_i T_{slot}}, \tag{6}$$

T_s is the average transmission time of a single packet, which depends on the physical layer (PHY) and MAC layer

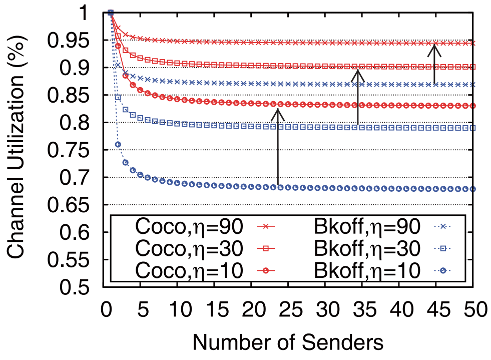


Fig. 4. Comparison between *Coco* and the backoff-based mechanism, with respect to channel utilization. *Coco* improves the achievable upper bound of channel utilization with different $\eta = T_c/T_{slot}$.

specifications. T_c is the average corruption time of a packet and T_{slot} is the time of a single slot. Note that in our model T_c is equal to T_s because transmissions of packets are aligned. T_{slot} is much shorter than T_s and T_c. For example in a typical implementation, we follow the standard of IEEE 802.15.4 in which the maximum packet size is 128 bytes and the maximum data rate is 250 kbps. Each slot lasts for about 0.032 ms, which is the transmission time of 1 byte. T_c lasts for at most 4.096 ms. Thus the ratio $\eta = T_c/T_{slot}$ is in the range from 1 to 128. The value of η matters as channel utilization is strongly related to packet length, which we will clarify in Fig. 4.

Since channel utilization is modeled as a function of p and the total number of senders, one can find the optimal probability p^{opt} that maximizes channel utilization (or minimize the inverse of Eqs. (6)). Therefore for a network with N senders, p_N^{opt} (the optimal transmitting probability with N senders) should be calculated as follows:

$$p_N^{opt} = \arg\max_p Util(p). \tag{7}$$

3.2 Improvement against Collision Avoidance

It's worth noticing that *Coco* improves the achievable upper bound of channel utilization, compared with conventional protocols. For backoff-based protocols, e.g., linear backoff in 802.15.4 and 802.11 Distributed Coordination Function (DCF) in 802.11 networks, the achievable upper bound of channel utilization is the ratio of successful slots to the total time. The performance gain brought by *Coco* is due to the significantly enhanced probability of a successful slot. Different from P_s, the probability $P_s^{'}$ for random backoff based protocols can be modeled as:

$$P_s^{'} = Np(1-p)^{N-1}. \tag{8}$$

In such protocols, senders take random backoffs and it is therefore difficult to calculate the duration of collisions T_c. Alternatively, we use T_s as the upper bound of T_c, similarly with the analysis in [19]. Fig. 4 shows the numerical comparison of achievable upper bound of channel utilization between *Coco* and backoff based protocols with different $\eta = T_c/T_{slot}$. It is clear that the gap is about 15 percent with $\eta = 10$, 10 percent with $\eta = 30$ and 8 percent with $\eta = 90$. Note that larger η brings higher channel utilization in average, and we will experimentally show the comparison of channel utilization with different packet lengths in the evaluation section.

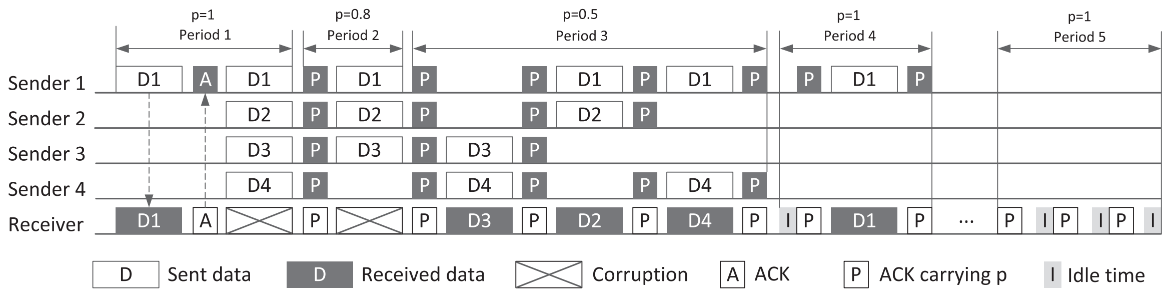


Fig. 5. Overview of *Coco*. Four senders transmit packets to a common receiver. *Coco* starts when a collision is detected (Period 1), and probability p is adjusted for best concurrency (Period 2, 3, 4). All senders transmit with the assigned p by the receiver. *Coco* ends when all senders finish their transmission (Period 5).

Note that the achievable channel utilization for random backoff based protocols seems as high as $86\,\text{percent}$ when $\eta = 90$. In practice, however, it is extremely difficult to achieve that performance. Our analysis above is based on the assumption that senders have the knowledge of each other and thus the optimal transmission probability p can be calculated. In fact, senders running random backoff protocol can only autonomously set their own backoff time to avoid collisions [20]. As a result, the global optimum cannot be reached with an increasing number of senders [21]. The authors in [8] reveal the performance degradation with a large number of senders for 802.11 DCF. In comparison, *Coco* is able to finely approach its optimal utilization via accurate estimation of network conditions and online regulation of the transmission probability p. In the next section, we show how we address the above design issues in the *Coco* protocol.

4 Design of *Coco*

Coco is a protocol for medium access control that coordinates the behavior of a receiver and multiple senders. Supported by the principle of collision tolerance, the basic idea of *Coco* is to align the packet transmissions from multiple concurrent senders and finely control the transmission probability p to achieve high channel utilization.

This section presents the design of *Coco*. We start with a brief overview of the workflow, and then explain the details of two key components: sender alignment and feedback control. We also analyze the protocol's performance optimality, convergence speed, fairness, and discuss its limitations.

4.1 Overview

Fig. 5 depicts the workflow of *Coco* with an example of four senders and a receiver. We assume each sender has one packet to send for simplicity. The whole workflow is divided into five periods. In period 1, only $S1$ (Sender 1) sends a packet. The receiver acknowledges $S1$ on receiving its packet. Because the ACK is broadcasted, not only $S1$, but also the other senders hear the ACK, from which they learn that the receiver is active and ready for receiving upcoming packets. Then they are triggered to transmit with the initial probability $p = 1$, which results in packet corruption. Seeing the corruption, the receiver realizes the channel is overly crowded and decides to reduce the transmission concurrency. So it regulates the probability p based on the feedback control algorithm (detailed in Section 4.3) and piggybacks the new value of $p = 0.8$ in the ACK packet.

In period 2, $S1$, $S2$ and $S3$ transmit with $p = 0.8$ and suffer from another corruption. As a result, the receiver further regulates p to 0.5. With the appropriately controlled probability, in period 3 $S3$, $S2$ and $S4$ transmit their packets successfully one after another. Note that transmissions turn out to be successful even when there are more than 1 concurrent transmissions.

$S3$, $S2$ and $S4$ finish transmission in period 3. Therefore in period 4, an idle slot occurs as p is too small in the case that only $S1$ is active. Under this condition, the receiver waits for a maximum interval T_{max} and realizes that the channel is under-utilized. So it resends a probe ACK after T_{max} with a new probability p, e.g., 1 finally. If there is no response after N_{max} ACKs ($N_{max} = 3$ in this example), the receiver believes all senders have finished transmissions and it may cease the ACK behavior, as shown in period 5.

In the following sections, we elaborate on the two components: (1) The mechanism for precise sender alignment and (2) The online feedback control algorithm to regulate p for concurrency control.

4.2 Align Senders' Packets

The first challenge mentioned in Section 2 can be overcome with the help of the ACK and the SFD (see Fig. 2) synchronization mechanism. In fact, the basic idea of utilizing ACK for synchronization has appeared in Fig. 5. Instead of transmitting randomly in traditional protocols, senders transmit by detecting the ACK packet broadcasted by the intended receiver. The principle is that when receiving an identical ACK packet, the SFD rising edge and falling edge of the ACK are strictly aligned for all senders. Therefore *Coco* utilizes the precise alignment of SFD edges for packets alignment. Specifically, all senders should respond to the receiver as soon as possible after receiving the ACK packet. Fig. 6 shows two senders are triggered by an ACK, their reception processes are precisely synchronized by the SFD interrupts (edges). With this design, the offset only occurs when senders respond the interrupt of SFD falling edge. In the evaluation part, we show that this delay is extremely

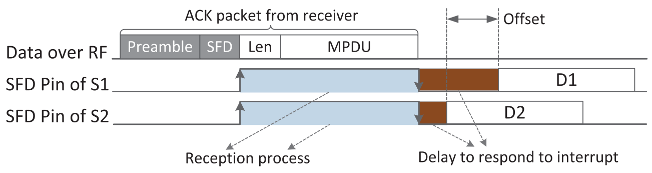


Fig. 6. Senders are aligned by SFD interrupts of the broadcasted ACK packet from the receiver. Transmission offsets are mainly caused by delay in responding to the falling-edge interrupt.

TABLE 1
Values of p^{opt} and P_c with N from 1 to 20

N	p^{opt}	P_c^{opt}	N	p^{opt}	P_c^{opt}
1	1	0	11	0.0351	0.0107
2	0.3533	0.0125	12	0.0320	0.0107
3	0.1621	0.0109	13	0.0294	0.0107
4	0.1105	0.0108	14	0.0272	0.0107
5	0.0843	0.0107	15	0.0253	0.0107
6	0.0683	0.0107	16	0.0237	0.0107
7	0.0574	0.0107	17	0.0222	0.0107
8	0.0495	0.0107	18	0.0209	0.0107
9	0.0436	0.0107	19	0.0198	0.0107
10	0.0389	0.0107	20	0.0188	0.0107

small. In this way, senders with the identical receiver are able to transmit their packets with bounded offset.

Exploiting ACK to trigger the transmission of senders also benefits resolution of hidden terminal problem [22], [23]. In *Coco*, senders explicitly contend. It makes no difference whether senders can sense the existence of each other or not. In evaluation part, we evaluate *Coco* in topologies with hidden terminals and demonstrate *Coco*'s robustness.

4.3 Feedback Control Algorithm

In Section 3, we analyze that p_N^{opt} can be numerically found. However, it is hard in practice, due to the following reasons. First, the receiver has no idea about the total number or the identities of senders. Thus it cannot directly calculate p_N^{opt} using Eqs. (6). Second, the total number and identities of active senders vary greatly over time, as some senders finish their transmission tasks earlier than the others and some senders may join in the concurrent transmissions without prior notice to the receiver. Considering the dynamics of networks, even there is a method to figure out the total number and identities of the senders, it still means considerable overhead at the receiver to reach the value of p_N^{opt}.

In order to obtain proper p in the dynamic network condition, we propose a feedback control algorithm based on the states of past slots. This algorithm releases *Coco* from the task of deciding the exact number of active senders and helps to obtain a close-to-optimal value of p_N^{opt} in real time. Next, we mainly answer the following questions: (1) How to judge whether p is proper or not? (2) If not, how to adjust p to approximate its optimal value? (3) What is the convergence speed and (4) What are the error bounds?

For the first question, we propose the following proposition:

Proposition 1. *P_c^{opt} means the optimal corruption probability when the channel utilization approximates its optimal value with number of senders N, and $\lim_{N\to\infty} P_c^{opt}$ is an ideal standard for deciding whether p is proper or not.*

As is shown in Table 1[2], P_c^{opt} converges to a constant value quickly. We see that P_c^{opt} is 0 when $N = 1$, while it approaches 0.0107 when N increases. In other words, the value of P_c^{opt} doesn't change when N increases 5. This property aids us to decide whether p is proper or not. We can

2. The values of p^{opt}, P_c^{opt} and P_s^{opt}, P_i^{opt} in Table 1 and Fig. 8 are from a numerical calculation based on the analysis in Section 3.

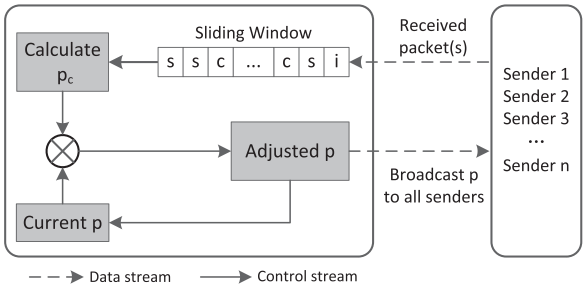


Fig. 7. The control diagram for a receiver. Receiver regards the received packet(s) as input and calculate P_c by counting the number of corrupted slots (n_c) in the sliding window with size W. The output is the adjusted probability p, which is carried in ACK and broadcasted to all senders.

first calculate P_c from the statistical number of corrupted slots in a time period and then compare P_c with $\lim_{N\to\infty} P_c^{opt}$, as is shown in Fig. 7. If P_c is larger than $\lim_{N\to\infty} P_c^{opt}$, which means that p is too large, we decrease p, otherwise we increase p. Note that when $N = 1$, P_c^{opt} is always 0, in which case we increase p until 1.

A natural question here is why we use the value of P_c^{opt} other than P_s^{opt} or P_i^{opt} (when N is infinite) as the standard to decide whether p is proper or not. In [19], the authors introduce the similar model and indeed look at the probability of idle slots i.e., P_i to decide how severe the contention is. The drawback, however, is that when the number of senders is small, e.g., less than 10, this method introduces significant errors. The reason is that P_s^{opt} and P_i^{opt} converge to constant values only when N approaches to a large number, e.g., 35 with $\eta = 10$, 30 with $\eta = 30$ and 20 with $\eta = 90$, as is shown in Fig. 8. So adjusting p based on the value of P_s^{opt} or P_i^{opt} is not accurate and results in great errors. While for P_c^{opt}, it approaches to its constant value quickly with the increase of N, e.g., 5, as is shown in Table 1.

After the current value of p is detected to be inappropriate, we apply a dichotomous algorithm to find the proper value of p for the current network condition. The pseudo code of this algorithm is shown in Algorithm 1. In this algorithm, p_l and p_u mean the lower and upper bound of the confidence interval. p is calculated as the center point of this interval. Initially $p_l = 0$, $p_u = 1$, and $p = 0.5$. In each iteration, the resulting P_c is compared with $\lim_{N\to\infty} P_c^{opt}$ to get the new interval bounds as well as the new p. Specifically, if P_c is smaller, the new interval is set to (p, p_u) and it is set to

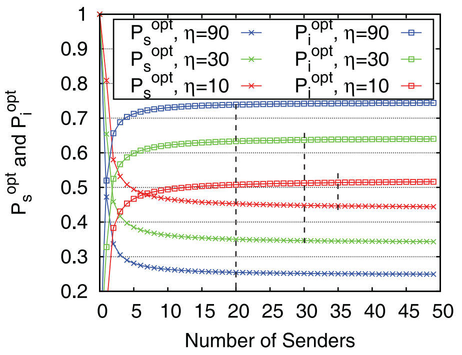


Fig. 8. Value of P_s^{opt} and P_i^{opt} when N changes from 1 to 50 with $\eta = 10$, 30 and 90. We can find that P_s^{opt} and P_i^{opt} converge slowly, compared with that of P_c^{opt} from Table 1.

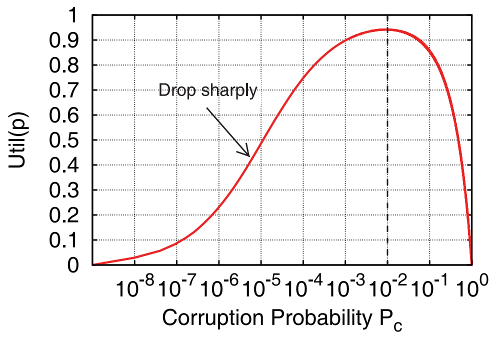


Fig. 9. The impact of P_c on $Util(p)$. On the left side of P_c^{opt}, $Util(p)$ drops quickly from the maximum to 0 while it changes much more slowly on the right side. (Note the log-scale of X-axis.)

(p_l, p) otherwise. Based on the new interval, p is further updated. The iterations do not halt until p converges to p^{opt}, for which we claim that P_c satisfies the following condition:

$$\lim_{N\to\infty} P_c^{opt} \le P_c < \lim_{N\to\infty} P_c^{opt} + \epsilon, \tag{9}$$

where ϵ is the error bound of P_c. Note that we restrict P_c to a range instead of the exact value, so as to avoid fluctuations in feedback control. Besides, the range is set to begin from $\lim_{N\to\infty} P_c^{opt}$ rather than from $\lim_{N\to\infty} P_c^{opt} - \epsilon$. The reason is that the value of $Util(p)$ varies greatly when P_c is in the range $(\lim_{N\to\infty} P_c^{opt} - \epsilon,\ \lim_{N\to\infty} P_c^{opt})$. Fig. 9 shows how $Util(p)$ changes with respect to P_c. It is easy to see that $Util(p)$ decreases to 0 quickly on the left side. Note that X-axis is log-scale.

Algorithm 1. The Feedback Control Algorithm

```
Input: $n_c$, $W$
Output: $p$
 1: $P_c \Leftarrow n_c/W$                          # Calculate $P_c$
 2: if $P_c^{opt} \le P_c < P_c^{opt} + \epsilon$ then
 3:     return $p$                                 # $p$ is proper
 4: else if $P_c < P_c^{opt}$ then
 5:     $p_l \Leftarrow p$
 6:     $p \Leftarrow (p_l + p_u)/2$               # Increase $p$
 7:     return $p$
 8: else if $P_c \ge P_c^{opt} + \epsilon$ then
 9:     $p_u \Leftarrow p$
10:     $p \Leftarrow (p_l + p_u)/2$               # Decrease $p$
11:     return $p$
12: end if
```

Errors introduced by ϵ. As P_c is not exactly equal to its optimal value, the real utilization function $Util(p)$ may not be able to achieve its optimal $Util^{opt}$ as shown in Fig. 4. To examine the error of $Util(p)$ caused by ϵ and choose a proper ϵ, we calculate the difference between $Util(p)$ and $Util^{opt}$ with four different ϵ values. Results are shown in Fig. 10. It shows that the error caused by ϵ is well bounded. For example, when $\epsilon = 0.03$, i.e., $0.0107 \le P_c < 0.0407$, the error of $Util(p)$ is limited to 3 percent. We choose $\epsilon = 0.05$ in our implementation.

Convergence speed. Another performance metric we care is the convergence speed of Algorithm 1. We claim that in this algorithm p can converge quickly from its initial value to the approximate value p^{opt}. From Table 1, we see that p^{opt}

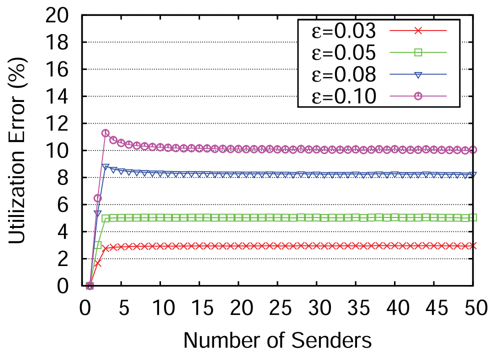


Fig. 10. Errors introduced by ϵ for a different number of senders. The errors are well bounded for different ϵ.

has a resolution of 0.001, when the total number of senders is at most 20. Therefore, the algorithm needs at most 10 iterations to regulate p from the initial value to p^{opt}. For a more intuitive understanding, we suppose for each iteration the receiver has to check states of 100 slots and each checking time lasts at most 4.096 ms. The total convergence time is at most $10 \times 100 \times 4.096$ ms, which is equal to 4.1 s.

4.4 Fairness of *Coco*

The essence of capture effect is that strong signal overwhelms the weak one(s). In this regard, it is unfair if a weak signal is always submerged by stronger ones. This unfairness problem of capture effect is discussed in [24] and [25]. We claim that *Coco* can mitigate this problem because of the mechanism to adjust the transmitting p dynamically.

The unfairness mentioned in the arts [24], [25] is due to the fact that the stronger senders always join in the contention, therefore the chance to transmit for the weak ones is grabbed by the stronger ones. While in *Coco*, it's is totally different. The key insight that helps *Coco* to achieve good fairness is the specially designed transmission mechanism. Transmitting with probability p grants all the senders sufficient randomness in the contention. Besides, by carefully choosing the transmitting probability p so that the expected number of senders in each slot $N * p$ is around 1. This not only makes the channel sufficiently used, but also guarantees the fairness that all the senders fairly share the channel. We conduct a set of experiments to validate the fairness of *Coco* and present the results in Section 5.

4.5 Discussion

As an MAC layer protocol, *Coco* can be a good candidate substitution of the traditional CSMA backoff mechanism for various protocol scenarios, such as data aggregation [26], [27], relay [28] and data collection [29], [30]. Especially when the data rate is high and nodes are dense [31], *Coco* shows it advantages. However, we claim that there are several issues for *Coco* to be considered.

Application scenarios. As transmission is triggered by the receiver's ACK, *Coco* is like a receiver-initiated protocol [32]. As a result, senders should transmit according to the coordination of the receiver. Another consideration is the tradeoff between random-backoff based protocols and *Coco*. Although backoff mechanism has poor performance when the number of senders is large, it shows flexibility and low complexity, especially when the number of senders is small. *Coco* has particular advantages in the scenarios of high contention and high data rates. When the networks are sparse (i.e., the neighbor size of a node is small) or the traffic load is low, the performance advantages of *Coco* might diminish. Also, *Coco* is suited for the applications with the goal of data collection efficiency rather than energy saving, as its design has extra energy consumption.

Application to OFDM 802.11 networks. The capture effect is a phenomenon associated with FM reception in which only the stronger of two signals at, or near, the same frequency or channel will be demodulated. In OFDM based 802.11 networks, there are several works [2], [33], [34] reporting the appearance of capture effect. Therefore, the collision tolerance capability is still applicable in OFDM based networks. Even so, we should pay attention to the application of capture effect in OFDM modulation based networks considering the slightly different frame structure. For example, the preamble field in OFDM frame has two parts, namely the short training sequences and the long ones, which is different from the DSSS frame.

Decoding more than one. As *Coco* follows the rule of simpleness and little modification on hardware, it decodes the mixture of multiple signals like the process with only one signal. The drawback for this is that it can only decode one packet in each reception. Existing work like [23], [35], [36], [37] can recover more than one signals in one round using more sophisticated techniques in the PHY layer. It's a good trial in the future to borrow the ideas of these work in decoding more than one signals in each round.

Why using a uniform p? In *Coco*, we utilize a uniform transmission probability p to maintain the simplicity and fairness property instead of differentiating nodes by their signal strength or distance and assigning them different p. First, calculation of the optimal value for any node k p_k^{opt} will be difficult considering the difference among nodes. The feedback control mechanism needs to take into account the status for each node, rather than a receiver-side view. Second, assigning nodes with different p result in unfairness.

5 Implementation and Evaluation

Implementation. We implement *Coco* in ContikiOS [15] on TelosB [16] platform. To facilitate the application of *Coco*, there are several issues in the implementation. Especially, we introduce the modifications to existing protocols and the receiver side logic. First, the protocol should be receiver-initiated instead of sender-initiated in order to achieve high synchronization accuracy with little overhead, as is stated in Section 4.2. Second, the carrier sensing function is disabled in *Coco* to allow the concurrent transmission of multiple senders. Third, for a receiver, the decoding and decision process is the same as the process with only one transmission, i.e., a packet can be correctly received when the PHY layer decoding and the MAC layer FCS (Frame Control Sequence) verification are both successful.

We conduct experiments to evaluate the performance of *Coco* from different aspects. First, we verify how well *Coco* copes with the challenges mentioned in Section 2 to achieve collision tolerance. Namely, we evaluate the synchronization accuracy and how well p is adjusted in practical networks. Second, we evaluate the robustness of *Coco* with different network settings. Third, we conduct

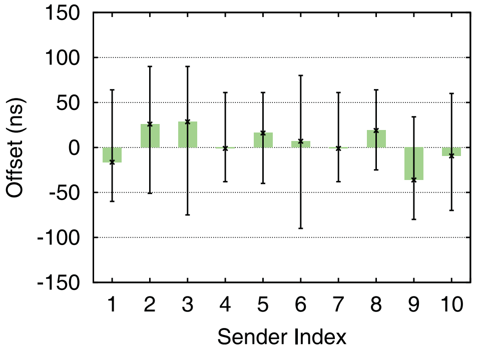


Fig. 11. Transmission offset among senders. The offset is measured between 10 different senders with respect to a reference sender on the receiver side.

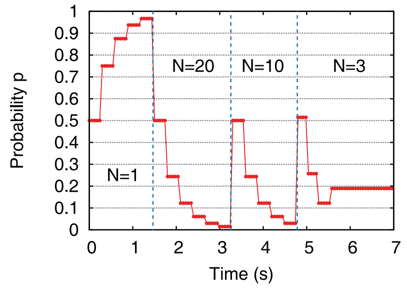
macro-benchmark experiments to evaluate the performance of *Coco* and we compare *Coco* with conventional protocols based on collision avoidance in terms of channel utilization. Then we investigate the energy consumption and evaluate the extra energy cost for *Coco*. Finally, we evaluate the fairness of *Coco*.

5.1 Timing Accuracy

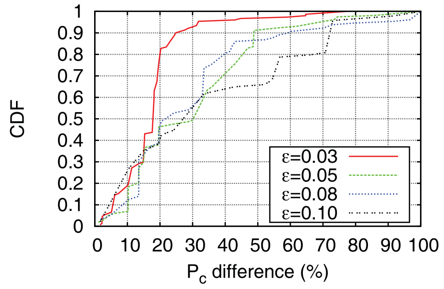
We first evaluate the time accuracy of aligning packets with ACK mechanism. As discussed in protocol design, senders are synchronized by SFD interrupts. In this experiment, we measure the delay in handling the interrupt, i.e., transmission offset, between two senders with a dual-channel digital oscilloscope. To eliminate the bias caused by hardware, we run this evaluation on 10 different senders. A reference node is placed 10 feet away from the receiver and the other 10 nodes are placed in a line, with distance ranging from 2 feet to 20 feet and node index increasing from 1 to 10. The offset is calculated with respect to the reference node. For each node, the experiments are repeated for 10 rounds. Fig. 11 shows the average, minimum and maximum offset for different nodes in the experiment.

Fig. 12a shows the convergence process. Initially, $p = 0.5$ and therefore it is not fit for the case $N = 1$. Thus p is increased from 0.5 to 0.75 and finally converges to 0.967. Then when N changes to 20, p is decreased and finally converges to a small value. When N changes from 20 to 10 and from 10 to 3, p adaptively changes and finally converges to a stable value. We can also calculate the convergence time in this figure. For example, when N increases from 1 to 20, the time for p to converge is about 1.7 s. While when N changes from 10 to 3, the time for convergence is only 0.7 s. This demonstrates that the feedback control algorithm can quickly adjust p according to the network condition.

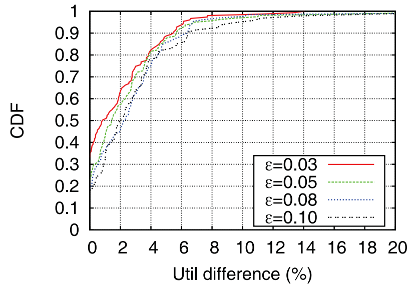
To further examine the impact of ϵ, we record P_c and $Util(p)$ and then compare them with the optimal values. We repeat the experiment with four different ϵ settings. The impact on P_c and $Util(p)$ introduced by ϵ are shown in Figs. 12b and 12c. The impact on P_c is calculated as the ratio of difference between P_c and P_c^{opt} to P_c^{opt}. The error of $Util$ is calculated in a similar way. From Fig. 12b, we can find that with smaller ϵ, the resulting error of P_c is also small. Fig. 12c shows us the utilization difference caused by a small ϵ is also small. Nevertheless, there is no significant difference among the different settings of ϵ. Especially, all four ϵ settings have less than 5 percent difference for 80 percent cases.



(a) Convergence process of p with $\epsilon = 0.05$.



(b) Difference between Pc and P_c^{opt}.



(c) Difference between $Util(p)$ and $Util^{opt}$.

Fig. 12. Evaluation of the feedback control algorithm. (a) depicts the dynamic adjustment process of p to approach p^{opt} with different N. (b) plots the difference between P_c and P_c^{opt} in the adjustment process with four settings of ϵ. (c) plots the difference between $Util(p)$ and $Util^{opt}$.

This tells us that as long as ϵ is small (e.g., 0.1), the difference between the achieved performance and the optimal performance is small.

5.2 Protocol Robustness Evaluation

In this section, we conduct a comprehensive evaluation on the robustness of *Coco* under various network settings. Especially we examine the impact from network environment, network topology and packet size. In all the following three experiments, we use 20 nodes transmitting packets to a common receiver. The transmitting power is still -5 dBm. Each sender transmits 100 packets and we repeat each test for 10 rounds to calculate statistical channel utilization.

5.2.1 Impact of Environment

Besides testbed-based experiments, we also conduct experiments in a hall and in an outdoor environment. Fig. 13 depicts the view of the three environments. For the outdoor scenario, we place nodes on the ground. In all three scenarios, we run *Coco* and record the corresponding channel utilization by looking at the successfully transmitted packets at the receiver side. Table 2 shows the average, minimum and maximum utilization in those environments. We find that there is no significant difference among those environments. The average utilization in an outdoor environment is slightly better than the other two cases. This experiment result shows that *Coco* can be applied to different environments.

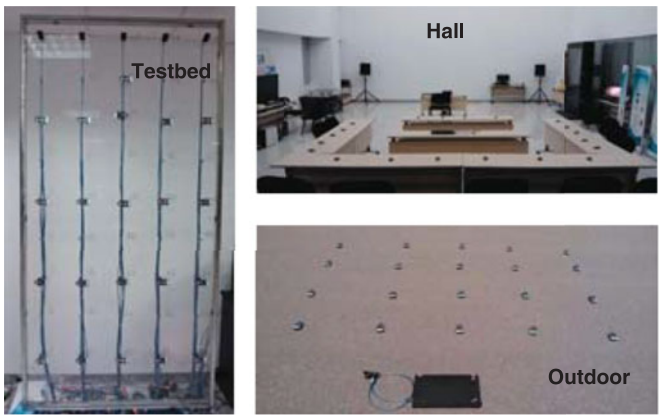


Fig. 13. Environments illustration. A testbed environment is shown in the left figure. We also evaluate *Coco* in an hall and nodes are placed on the desks, which is shown in the top-right figure. The bottom-right figure shows the an outdoor environment and nodes are placed on the ground.

5.2.2 Impact of Packet Length

The second parameter we examine is the average packet length. As discussed in the protocol design, the upper bound of optimal utilization is related to $\eta = T_c/T_{slot}$. Packet length l is set to 20, 60 and 100 bytes respectively and we record the corresponding utilization in the testbed environment. For different packet lengths, we also evaluate the utilization when there exist hidden terminal. The results are shown in Figs. 14a and 14b. We can see that for both cases, *Coco* achieves higher utilization than collision avoidance based approaches. Further, no matter there is hidden terminal or not, *Coco* achieves higher utilization for longer packets. Details are talked in the overall evaluation.

5.2.3 Impact of Topology

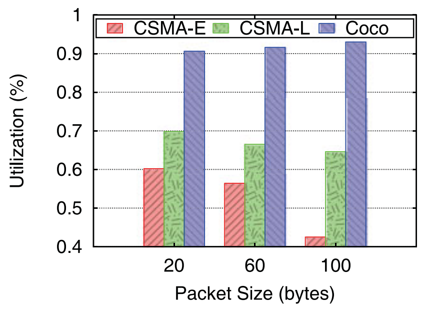
The last parameter that may affect the performance of *Coco* is network topology. We place nodes to form three different topologies, namely (1) the *circle topology* where senders are placed to form a circle around the receiver, (2) the *line topology* where senders are placed in a line and the receiver is located at one side of the line and (3) the *random topology* in which senders are randomly placed near the receiver. In circle topology, nodes cannot sense the existence of the nodes on the opposite side, resulting in hidden terminals in the network. In line topology, nodes far away from the receiver have weaker received signal strength than nodes that are closer to the receiver. The circle topology is used to test the performance of *Coco* in face of hidden terminal while the line topology is to test *Coco*'s performance for nodes with different signal strength. We introduce the results of this set of experiments in the following section.

5.3 Overall Performance Evaluation

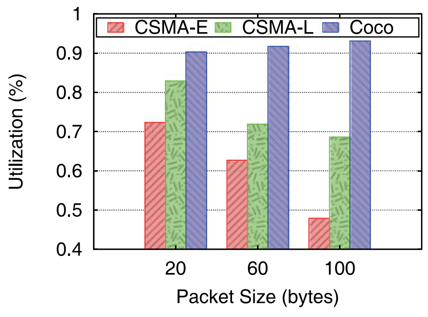
We compare *Coco* with other existing random-backoff based protocols to show the performance gain. In 802.15.4 networks, we choose B-MAC [38] as the representative of random-backoff based protocols.

TABLE 2
Performance in Three Environments

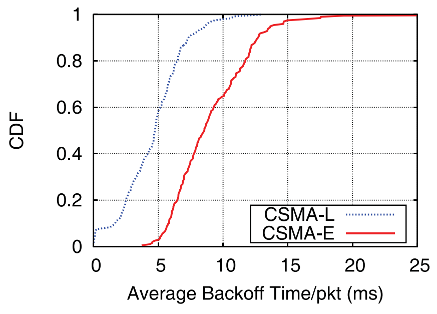
Env.	*avg.*	*min.*	*max.*
Testbed	0.904	0.802	0.968
Hall	0.901	0.798	0.978
Outdoor	0.915	0.802	0.977



(a) With Hidden terminal.



(b) Without Hidden Terminal.



(c) Per-packet backoff time.

Fig. 14. Performance comparison with CSMA linear backoff (CSMA-L) and CSMA exponential backoff (CSMA-E) with different packet lengths. (a) shows the results in a scenario with hidden terminals (circle topology) and (b) shows the scenario without hidden terminals. (c) shows the CDF of average backoff time per packet in both CSMA-E and CSMA-L.

5.3.1 Comparison with CSMA Backoff

In this experiment, we implement both linear and exponential backoff in B-MAC, denoted as CSMA-L and CSMA-E respectively. Experiments are conducted in both circle topology (with hidden terminals) and random topology with packet lengths of 20, 60 and 100 bytes. We measure the time for all senders to finish transmission at the receiver side to compute the overall utilization. The transmitting power of nodes in both protocols is set to -5 dBm.

Figs. 14a and 14b show the results. There are several findings to be revealed in this experiment. First, CSMA protocols are not robust to hidden terminals. Both CSMA-L and CSMA-E demonstrate performance degradation in circle topology with hidden terminals. Second, the impact of packet length on CSMA protocols and *Coco* are essentially different. For CSMA-L and CSMA-E, shorter packets are preferred. While for *Coco*, longer packets are better. The reason lies in the fact that (1) packet length doesn't affect the collision probability in *Coco*, while it does for CSMA protocols (long packets result in more backoff), and (2) longer packets benefit more for *Coco* than CSMA protocols. Third, CSMA-L shows advantages against CSMA-E in 802.15.4 networks. The exponential backoff scheme does not perform well in 802.15.4 networks as it does in 802.11 networks. The reason is that the maximum packet length is only 128 bytes in 802.15.4 networks while it is 2,304 bytes in 802.11 networks. Thus, it is not desirable to apply larger backoff window in 802.15.4 networks. Fig. 14c clearly shows the per-lacket backoff time for CSMA-E and CSMA-L. At last, the overall utilization of *Coco* outperforms both CSMA-L and CSMA-E in both topologies with different packet lengths. The performance improvement ranges from about 10 percent to about 50 percent.

To further investigate the performance of CSMA backoff protocols and *Coco*, we look into the backoff time. Fig. 14c shows the average backoff time for each packet in the experiment. We find that the average backoff time for each packet is at least 5 ms for CSMA-E. The average backoff time of CSMA-L for each packet is about 7 ms for 80 percent packets. However, backoff time is not required in *Coco*. Therefore, it is easy to understand the reason for different performances of CSMA-E, CSMA-L and *Coco*.

5.3.2 Improvement to AMAC

Besides comparison with CSMA protocols that run in an always-on mode, we show that *Coco* can also be applied in duty-cycling wireless sensor networks (WSNs). Here we select AMAC [3], which is the most representative receiver-initiated duty-cycling protocol in TinyOS. The basic mechanism of AMAC works as follows. Each node periodically wakes up. After waking up, a potential receiver polls the channel by sending polling packets to see if there is any transmission to itself. If a sender has packets to transmit, it will sense the channel to see if there are any polling packets from the receiver. If there is, the sender will send packets to the receiver.

As stated in [3], AMAC suffers from collisions in dense networks. In AMAC, senders have asynchronous wake-up schedules. It is possible that multiple senders for the same receiver wake up in the same cycle. Those senders will receive the same probe from the receiver and then they will perform backoffs to avoid collisions. As in CSMA, collisions occur when two senders choose the same backoff time.

In our experiment, we integrate *Coco* with AMAC to improve its ability to handle collision. We piggyback the probability p in the polling packets. The modification is that after receiving a probe packet from the receiver, all senders who are awake should perform backoff in a time window of less than 160 μs, as is the time of preamble plus SFD byte. After the backoff timer expires, senders apply p to transmit. This backoff mechanism ensures that backoff time is bounded in AMAC. We record the time that a packet needs to wait (including backoff and duty-cycling period) before and after *Coco* is added into AMAC.

As shown in Fig. 15, after integrating *Coco* with AMAC, the average used time for each packet is significantly reduced. The average used time per packet is reduced from

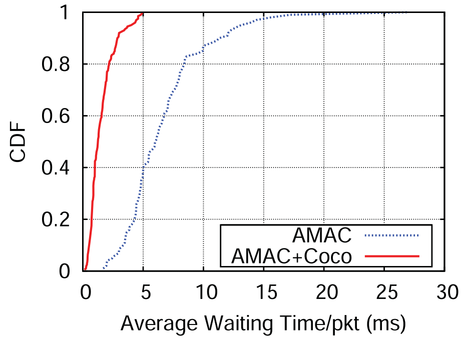


Fig. 15. Per-packet waiting time before and after AMAC is integrated with *Coco*.

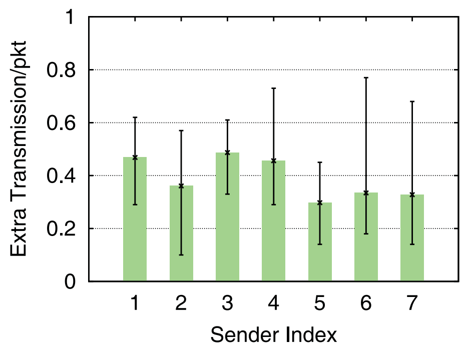


Fig. 16. The number of extra transmissions per packet in *Coco* for a 7-sender case.

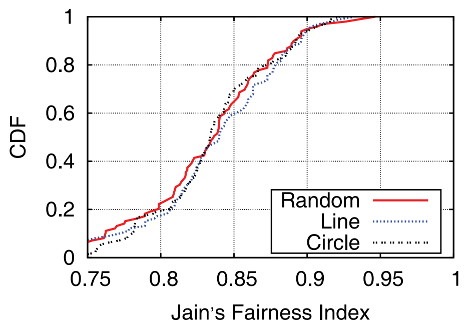


Fig. 17. Distribution of Jain's Fairness Index for all senders in random, line and circle topologies.

about 7 ms to only 1 ms with *Coco*. This indicates *Coco* is efficient to help resolve collisions in AMAC.

5.4 Extra Energy Cost Evaluation

As discussed above, *Coco* encourages concurrent transmissions. This inevitably introduces extra energy cost as extra senders transmit in a slot while their packets are not received. In this evaluation, we quantify the extra energy cost in terms of the number of extra transmissions for each sender considering the fact the extra energy cannot be measured directly.

In this experiment, we use seven senders and one receiver. We maintain one counter on each sender and seven counters on the receiver. The counter on the sender side records the number of packets it has transmitted while the counters at the receiver side record the number of packets received from each sender. The number of extra packets transmitted by each sender can be calculated as the difference of the two counters. For simplicity, we don't consider retransmission and think that the links are stable and there is no packet lost. In this experiment, each sender is asked to transmit 100 packets. We divide 100 by the number of extra transmitted packets to get the number of extra transmissions per packet. Fig. 16 shows the results. From this figure, we find that the average number of extra transmissions per packet is less than 0.5, which means that each successful packet is along with at most 0.5 extra transmissions in average.

5.5 Fairness Evaluation

Finally, we conduct experiments to evaluate fairness mentioned in Section 4 in the aforementioned three topologies. Here fairness is considered as the long-term share of channel utilization. We use 10 senders to transmit to a common receiver, and measure both the individual utilization of each sender and the overall utilization on the receiver side. We use two test scenarios: (1) 10 senders concurrently transmit to a receiver and (2) the number of senders is dynamically changing. In both scenarios, the transmitting power is set to -5 dBm.

Fig. 17 shows the distribution of Jain's Fairness Index for the first scenario in the three topologies. We find that the fairness index for all the three topologies is acceptable. The mean value of fairness index for the three topologies is about 0.83. Fig. 18 is a stream graph that depicts the results in the second scenario. In this situation, the number of active senders changes dynamically. At the beginning, there is only one active sender ($S1$). Then, all 10 senders ($S1$-$S10$) become active. Then the number of active senders decreases gradually and finally only $S1$ is active. We can see that at the instant when all senders become active, *Coco* accordingly reduces the share of $S1$ and allocates it to other newly-joined nodes. After that, as the number of active senders decreases, *Coco* evenly divides the available bandwidth among the remaining active senders. This experiment shows that in *Coco*, senders can share the channel in a dynamic environment in a fair way.

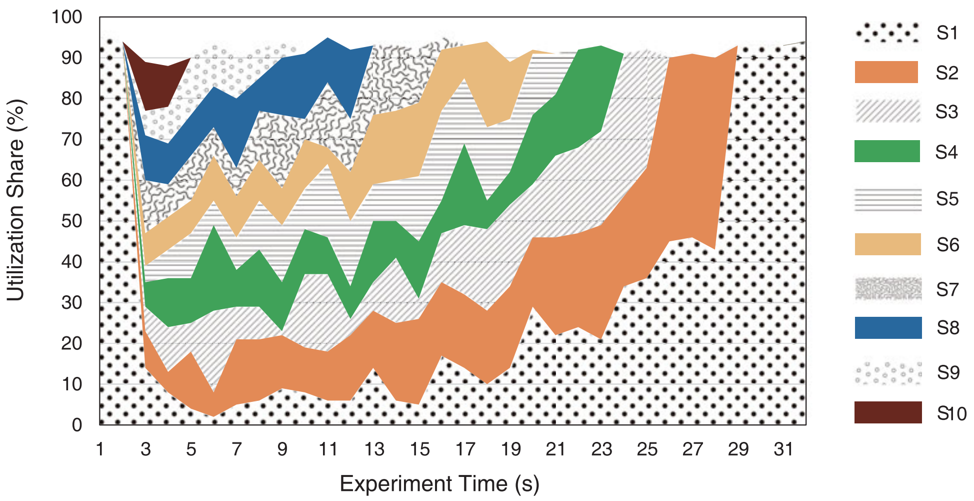


Fig. 18. The stream graph depicts the dynamic channel utilization over time. From the view of receiver, channel utilization can be as high as above 90 percent in most of the time. From the view of senders, they can fairly share the channel despite of the changes of the total number of active senders.

6 Related Work

Collision resolution protocols for improving channel utilization fall into the following three categories:

Collision avoidance. This is the most typical class of protocols [4]. Based on the mechanism how to avoid collisions, collision avoidance based protocols can be divided into schedule based and contention based. For the former, TDMA and FDMA are the representative protocols. Active senders are coordinated and allocated with different time slots/frequencies for transmission [39], [40]. For the latter, CSMA is applied as the medium access control method and senders perform random backoff for collision avoidance [41]. The overhead in avoiding collisions, however, degrades the performance of channel utilization. For example, coordination and synchronization in schedule based protocols consumes a large portion of transmission time. Contention based protocols conservatively perform backoff and result in many idle slots that cannot be utilized.

Collision tolerance. Different from collision avoidance, the idea of collision tolerance allows collisions. Flash flooding [42], Chorus [43] and Glossy [44] propose protocols for efficient data transmission exploiting capture effect in flooding scenarios. However, the common limitation of the existing protocols trying to use collision tolerance is that they can be only applied in flooding or broadcasting scenarios, where transmitted packets must carry the same data. This requirement greatly limits their application scope [45]. The behind reason is that these protocols fail to understand the basic timing and concurrency requirements of collision tolerance, which are investigated in this paper.

Protocols with PHY layer techniques. Another category of protocols try to recover collided signals with advanced PHY layer techniques [23], [35], [46], [47], [48], [49]. For example, Zigzag [23] iteratively decodes the collision-free part in the collided signals first and then subtracts it from the collided signal. SIC [35] on the other hand, recovers the strong signal with capture effect and then recovers the weak signal by cancelling the strong one from the collided signals. Also in recent years, there is a thread of protocols which rely on the property of enterprise WLANs for collision recovery, in order to improve the throughput. The representative work are like [36], [37], [50], [51]. These protocols require specific modification in the PHY layer and are not supported in commercial hardware. On the contrary, *Coco* is light-weight. It needs little modification in the PHY layer and can be directly used on off-the-shelf hardwares.

7 Conclusion and Future Work

Collision resolution is a crucial issue in wireless networks. Based on the insight of collision tolerance, this paper proposes *Coco*, a novel practice of MAC protocol that exploits the opportunities to transmit packets under collisions. *Coco* addresses the practical challenges in achieving collision tolerance and brings significant performance gain in wireless channel utilization. We implement *Coco* in 802.15.4 networks and evaluate its performance in various network settings. The evaluation results demonstrate that *Coco* significantly improves channel utilization.

In our future work, we are going to integrate *Coco* with other application-layer protocols, e.g., the CTP [52] protocol in TinyOS. Moreover, we plan to extend the implementation and evaluation of *Coco* to 802.11 networks and extend the application scenario of *Coco* from single receiver to multiple-receiver case.

Acknowledgments

This study is supported in part by HKRGC under grants GRF-621413, GRF-16211614, National Basic Research Program (973 program) under Grant of 2014CB347800, National Natural Science Fund for Excellent Young Scientist No. 61422207, China NSFC Grant 61572277, 61529202, 61532012, 61472259, Shenzhen Science and Technology Foundation (No. KQCX20150324160536457), and Guangdong Young Talent Project 2014TQ01X238.

References

[1] Y. C. Tay, K. Jamieson, and H. Balakrishnan, "Collision-minimizing CSMA and its applications to wireless sensor networks," *IEEE J. Sel. Areas Commun.*, vol. 22, no. 6, pp. 1048–1057, 2004.

[2] K. Whitehouse, A. Woo, F. Jiang, J. Polastre, and D. Culler, "Exploiting the capture effect for collision detection and recovery," in *Proc. IEEE Workshop Embedded Netw. Sensors*, 2005, pp. 45–52.

[3] P. Dutta, S. Dawson-Haggerty, Y. Chen, C.-J. M. Liang, and A. Terzis, "Design and evaluation of a versatile and efficient receiver-initiated link layer for low-power wireless," in *Proc. 8th ACM Conf. Embedded Netw. Sensor Syst.*, 2010, pp. 1–14.

[4] Y. Z. Zhao, C. Miao, M. Ma, J. B. Zhang, and C. Leung, "A survey and projection on medium access control protocols for wireless sensor networks," *ACM Comput. Survey*, vol. 45, no. 1, pp. 7: pp. 1–7:37, 2012.

[5] Y. Liu, Y. He, M. Li, J. Wang, K. Liu, and X. Li, "Does wireless sensor network scale? A measurement study on greenorbs," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 10, pp. 1983–1993, Oct. 2013.

[6] Y. Liu, M. Liu, and J. Deng, "Evaluating opportunistic multi-channel MAC: Is diversity gain worth the pain?" *IEEE J. Sel. Areas Commun.*, vol. 31, no. 11, pp. 2301–2311, Nov. 2013.

[7] H. Li, K. Wu, Q. Zhang, and L. M. Ni, "Cuts: Improving channel utilization in both time and spatial domain in WLANS," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 6, pp. 1413–1423, Jun. 2014.

[8] G. Bianchi, "Performance analysis of the ieee 802.11 distributed coordination function," *IEEE J. Sel. Areas Commun.*, vol. 18, no. 3, pp. 535–547, Mar. 2000.

[9] LAN/MAN Standards Committee, "IEEE Standford - Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs)," *IEEE Comput. Soc.*, 2003.

[10] LAN/MAN Standards Committee, "IEEE Standford - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," *IEEE Comput. Soc.*, 2012.

[11] X. Ji, Y. He, J. Wang, W. Dong, X. Wu, and Y. Liu, "Walking down the stairs: Efficient collision resolution for wireless sensor networks," in *Proc. IEEE Int. Conf. Comput. Commun.*, 2014, pp. 961–969.

[12] K. Leentvaar and J. Flint, "The capture effect in FM receivers," *IEEE Trans. Commun.*, vol. 24, no. 5, pp. 531–539, May 1976.

[13] X. Ji, Y. He, J. Wang, K. Wu, K. Yi, and Y. Liu, "Voice over the dins: Improving wireless channel utilization with collision tolerance," in *Proc. IEEE Int. Conf. Netw. Protocols*, 2013, pp. 1–10.

[14] K. Wu, H. Tan, Y. Liu, J. Zhang, Q. Zhang, and L. M. Ni, "Side channel: Bits over interference," *IEEE Trans. Mobile Comput.*, vol. 11, no. 8, pp. 1317–1330, Aug. 2012.

[15] A. Dunkels, B. Grönvall, and T. Voigt, "Contiki-a lightweight and flexible operating system for tiny networked sensors," *Local Comput. Netw., 29th Annu. IEEE Int. Conf.*, pp. 455–462, 2004.

[16] J. Polastre, R. Szewczyk, and D. Culler, "Telos: Enabling ultra-low power wireless research," in *Proc. ACM 4th Int. Symp. Inform. Process. Sensor Netw.*, 2005, pp. 364–369.

[17] M. Zorzi and R. Rao, "Capture and retransmission control in mobile radio," *IEEE J. Sel. Areas. Commun.*, vol. 12, no. 8, pp. 1289–1298, Oct. 1994.

[18] A. Krishna and R. O. LaMaire, "A comparison of radio capture models and their effect on wireless LAN protocols," in *Proc. IEEE 3rd Annu. Int. Conf. Univ. Personal Commun.*, 1994, pp. 666–672.

[19] M. Heusse, F. Rousseau, R. Guillier, and A. Duda, "Idle sense: An optimal access method for high throughput and fairness in rate diverse wireless lans," in *Proc. ACM Conf. Appl. Technol. Archit. Protocols Comput. Commun.*, 2005, pp. 121–132.

[20] Z. Lu, W. Wang, and C. Wang, "Modeling and performance evaluation of backoff misbehaving nodes in CSMA/CA networks," *IEEE Trans. Mobile Comput.*, vol. 11, no. 8, pp. 1331–1344, Aug. 2012.

[21] C. Jiang, Y. Shi, Y. Hou, W. Lou, S. Kompella, and S. Midkiff, "Toward simple criteria to establish capacity scaling laws for wireless networks," in *Proc. IEEE INFOCOM*, 2012, pp. 774–782.

[22] E. Magistretti, O. Gurewitz, and E. W. Knightly, "802.11ec: Collision avoidance without control messages," in *Proc. ACM 18th Annu. Int. Conf. Mobile Comput. Netw.*, 2012, pp. 65–76.

[23] S. Gollakota and D. Katabi, "Zigzag decoding: Combating hidden terminals in wireless networks," in *Proc. ACM SIGCOMM Conf. Data Commun.*, 2008, pp. 159–170.

[24] H. Chang, V. Misra, and D. Rubenstein, "Fairness and physical layer capture in random access networks," in *Proc. 4th Annu. IEEE Commun. Soc. Conf. Sensor Mesh Ad Hoc Commun. Netw.*, 2007, pp. 381–390.

[25] C. Ware, J. Judge, J. Chicharo, and E. Dutkiewicz, "Unfairness and capture behaviour in 802.11 AdHoc networks," in *Proc. IEEE Int. Conf. Commun.*, 2000, pp. 159–163.

[26] L. Yu, J. Li, S. Cheng, S. Xiong, and H. Shen, "Secure continuous aggregation in wireless sensor networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 3, pp. 762–774, Mar. 2014.

[27] X. Zheng, J. Wang, W. Dong, Y. He, and Y. Liu, "Bulk data dissemination in wireless sensor networks: Analysis, implications and improvement," *IEEE Trans. Comput.*, vol. 65, no. 5, pp. 1428–1439, May 2016.

[28] L. Guo, X. Ding, H. Wang, Q. Li, S. Chen, and X. Zhang, "Cooperative relay service in a wireless LAN," *IEEE J. Sel. Areas Commun.*, vol. 25, no. 2, pp. 355–368, Feb. 2007.

[29] X. Ji, J. Wang, M. Liu, Y. Yan, P. Yang, and Y. Liu, "Hitchhike: Riding control on preambles," in *Proc. IEEE Int. Conf. Comput. Commun.*, 2014, pp. 2499–2507.

[30] L. Wang and W. Liu, "Navigability and reachability index for emergency navigation systems using wireless sensor networks," *Tsinghua Sci. Tech.*, vol. 16, no. 6, pp. 657–668, 2011.

[31] K. Jamieson, H. Balakrishnan, and Y. C. Tay, "Sift: A MAC protocol for event-driven wireless sensor networks," in *Proc. 3rd Eur. Conf. Wireless Sensor Netw.*, 2006, pp. 260–275.

[32] Y. Sun, O. Gurewitz, and D. B. Johnson, "Ri-Mac: A receiver-initiated asynchronous duty cycle MAC protocol for dynamic traffic loads in wireless sensor networks," in *Proc. 6th ACM Conf. Embedded Netw. Sensor Syst.*, 2008, pp. 1–14.

[33] J. Lee, W. Kim, S.-J. Lee, D. Jo, J. Ryu, T. Kwon, and Y. Choi, "An experimental study on the capture effect in 802.11 a networks," in *Proc. 2nd ACM Int. Workshop Wireless Netw. Testbeds Experimental Evaluation Characterization*, 2007, pp. 19–26.

[34] C. Ware, J. Chicharo, and T. Wysocki, "Modelling of capture behaviour in IEEE 802.11 radio modems," in *IEEE Int. Conf. Telecommun.*, 2001.

[35] D. Halperin, T. Anderson, and D. Wetherall, "Taking the sting out of carrier sense: Interference cancellation for wireless lans," in *Proc. ACM Int. Conf. Mobile Comput. Netw.*, 2008, pp. 339–350.

[36] W. Zhou, T. Bansal, P. Sinha, and K. Srinivasan, "Bbn: Throughput scaling in dense enterprise WLANS with bind beamforming and nulling," in *Proc. ACM 20th Annu. Int. Conf. Mobile Comput. Netw.*, 2014, pp. 165–176.

[37] T. Bansal, B. Chen, P. Sinha, and K. Srinivasan, "Symphony: Cooperative packet recovery over the wired backbone in enterprise WLANS," in *Proc. ACM 19th Annu. Int. Conf. Mobile Comput. Netw.*, 2013, pp. 351–362.

[38] J. Polastre, J. Hill, and D. Culler, "Versatile low power media access for wireless sensor networks," in *Proc. ACM 2nd Int. Conf. Embedded Netw. Sensor Syst.*, 2004, pp. 95–107.

[39] S. Sen, R. Roy Choudhury, and S. Nelakuditi, "No time to countdown: Migrating backoff to the frequency domain," in *Proc. ACM 17th Annu. Int. Conf. Mobile Comput. Netw.*, 2011, pp. 241–252.

[40] L. Wang, K. Wu, J. Xiao, and M. Hamdi, "Harnessing frequency domain for cooperative sensing and multi-channel contention in CRAHNs," *IEEE Trans. Wireless Commun.*, vol. 13, no. 1, pp. 440–449, Jan. 2014.

[41] P. Yang, B. Li, J. Wang, X. Y. Li, Z. Du, Y. Yan, and Y. Xiong, "Online sequential channel accessing control: A double exploration vs. exploitation problem," *IEEE Trans. Wireless Commun.*, vol. 14, no. 8, pp. 4654–4666, Aug. 2015.

[42] J. Lu and K. Whitehouse, "Flash flooding: Exploiting the capture effect for rapid flooding in wireless sensor networks," in *Proc. IEEE INFOCOM*, 2009, pp. 2491–2499.

[43] X. Zhang and K. Shin, "Chorus: Collision resolution for efficient wireless broadcast," in *Proc. IEEE 29th Conf. Inform. Commun.*, 2010, pp. 1747–1755.

[44] F. Ferrari, M. Zimmerling, L. Thiele, and O. Saukh, "Efficient network flooding and time synchronization with glossy," in *Proc. 10th Int. Conf. Inform. Process. Sensor Netw.*, 2011, pp. 73–84.

[45] R. Tan, G. Xing, J. Wang, and H. C. So, "Exploiting reactive mobility for collaborative target detection in wireless sensor networks," *IEEE Trans. Mobile Comput.*, vol. 9, no. 3, pp. 317–332, 2010.

[46] S. Sen, R. Roy Choudhury, and S. Nelakuditi, "CSMA/CN: Carrier sense multiple access with collision notification," in *Proc. 16th Annu. Int. Conf. Mobile Comput. Netw.*, 2010, pp. 25–36.

[47] Y. Yan, P. Yang, X. Li, Y. Tao, L. Zhang, and L. You, "Zimo: Building cross-technology MIMO to harmonize zigbee smog with WiFi flash without intervention," in *Proc. 19th Annu. Int. Conf. Mobile Comput. Netw.*, 2013, pp. 465–476.

[48] H. Zeng, Y. Shi, Y. T. Hou, W. Lou, S. Kompella, and S. F. Midkiff, "An analytical model for interference alignment in multi-hop mimo networks," *IEEE Trans. Mobile Comput.*, vol. 15, no. 1, pp. 17–31, Jan. 2016.

[49] P. Yang, Y. Yan, X. Y. Li, Y. Zhang, Y. Tao, and L. You, "Taming cross-technology interference for Wi-Fi and ZigBee coexistence networks," *IEEE Trans. Mobile Comput.*, vol. 15, no. 4, pp. 1009–1021, Apr. 2016.

[50] G. R. Woo, P. Kheradpour, D. Shen, and D. Katabi, "Beyond the bits: Cooperative packet recovery using physical layer information," in *Proc. 13th Annu. ACM Int. Conf. Mobile Comput. Netw.*, 2007, pp. 147–158.

[51] R. Murty, J. Padhye, R. Chandra, A. Wolman, and B. Zill, "Designing high performance enterprise Wi-Fi networks." in *Proc. 5th USENIX Symp. Netw. Syst. Des. Implementation*, 2008, pp. 73–88.

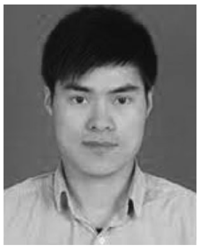
[52] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis, "Collection tree protocol," in *Proc. ACM SenSys*, 2009, pp. 1–14.

Xiaoyu Ji received the BS degree in electronic information & technology and instrumentation science from Zhejiang University, Hangzhou, China, in 2010. He received the PhD degree from the Department of Computer Science, Hong Kong University of Science and Technology, in 2015. He is currently with the Huawei Future Networking Theory Lab in Hong Kong. His research interests include protocol design in wireless networks, especially with cross-layer techniques, network measurement, and human computer interaction. He is a member of IEEE.

Yuan He received the BE degree from the University of Science and Technology of China, the ME degree from the Institute of Software, Chinese Academy of Sciences, and the PhD degree from the Hong Kong University of Science and Technology. He is currently an associate professor in the School of Software and TNLIST, Tsinghua University. His research interests include wireless networks, Internet of Things, and mobile & pervasive computing. He is a member of the IEEE and ACM.

Jiliang Wang received the BE degree in computer science and technology from the University of Science and Technology of China and the PhD degree in computer science and engineering from the Hong Kong University of Science and Technology, respectively. He is currently an assistant professor in the School of Software, Tsinghua University. His research interests include wireless sensor networks, network measurement, and mobile networks. He is a member of the IEEE.

Kaishun Wu received the PhD degree in computer science and engineering from the Hong Kong University of Science and Technology, in 2011. After that, he worked as a research assistant professor in the Hong Kong University of Science and Technology. From 2013, he joined Shenzhen University as a distinguished professor. He has co-authored two books and published more than 80 refereed papers in international leading journals and primer conferences. He is the inventor of six US and 45 Chinese pending patents (14 are issued). He received the Best Paper Awards at IEEE Globecom 2012, IEEE ICPADS 2012 and IEEE MASS 2014. He received 2014 IEEE ComSoc Asia-Pacic Outstanding Young Researcher Award and was selected as 1000 Talent Plan for Young Researchers.



Daibo Liu received the BE degree in computer science and technology from Dalian Maritime University, Liaoning, China, in 2009, and the ME degree in computer science and engineering from the University of Electronic Science and Technology of China, Chengdu, China, in 2012. He is currently working toward the PhD degree at the University of Electronic Science and Technology of China. His current research interest is wireless sensor networks. He is a member of the IEEE.

Ke Yi received the BE degree from Tsinghua University and the PhD degree from Duke University, in 2001 and 2006, respectively, both in computer science. He is currently an associate professor at the Hong Kong University of Science and Technology. He belongs to both the Theoretical Computer Science Group and the Database Group at HKUST. His research interests include algorithms on big data, databases, data summarization, algorithms on distributed data, data streams, external memory algorithms, data structures, and computational geometry. He is a senior member of the IEEE.

Yunhao Liu received the BS degree from the Automation Department, Tsinghua University, and the MA degree from Beijing Foreign Studies University, China. He received the MS and PhD degrees in computer science and engineering from Michigan State University. He is currently the dean of the School of Software, and holds ChangJiang Chair Professorship at Tsinghua University. He is serving as the associate editor-in-chief for the *IEEE Transactions on Parallel and Distributed Systems* and associate editor for the *IEEE/ACM Transactions on Networking* and *ACM Transactions on Sensor Networks*. He is a fellow of the IEEE (Citations: for contributions to Wireless Sensor Network and Systems), an ACM Distinguished Speaker, and chair for the ACM China Council. His research interests include sensor network and iot, localization, network diagnosis, rfid, distributed systems, and cloud computing.