Voice over the Dins: Improving Wireless Channel Utilization with Collision Tolerance

Xiaoyu Ji*†, Yuan He†, Jiliang Wang†, Kaishun Wu‡, Ke Yi*, Yunhao Liu†
* Department of Computer Science and Engineering, Hong Kong University of Science and Technology
† School of Software, TNLIST, Tsinghua University
‡ GuangZhou HKUST Fok Ying Tung Research Institute
Email: xji@cse.ust.hk, {he, jiliang}@greenorbs.com, {kwinson, yike}@cse.ust.hk, yunhao@greenorbs.com

***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 MAC 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 fine control of the 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% in general cases, compared with state-of-the-arts protocols.**

I. Introduction

Wireless networks suffer from collisions. This is essentially due to the broadcast nature of wireless communications. Simultaneous transmissions that collide in a common channel are likely to interfere with each other and thus cause corruptions of the transmitted packets. Without appropriate handling of collisions, network performance like channel utilization is degraded. 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 collision. 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. 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 [1].

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[1] 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 surprising result is that 5 more packets are successfully transmitted using collision tolerance. The utilization is enhanced by 71.4%.

The feasibility of collision tolerance actually comes from the physical layer implementations of wireless networks (e.g., 802.15.4 and 802.11 networks). For example, the DSSS (direct-sequence spread spectrum) modulation scheme in those implementations increase the resistance to interference by introducing redundancy [2]. 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.

In order to address the above challenges, in this paper we propose *Coco*, a protocol for medium access control that leverages the ability of collision tolerance. Using *Coco*, concurrent senders optimistically contend with each other to access the shared medium. A feedback control mechanism is devised to assign an appropriate transmission probability p for colliding senders and all senders transmit with the probability

[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.

978-1-4799-1270-4/13/$31.00 ©2013 IEEE

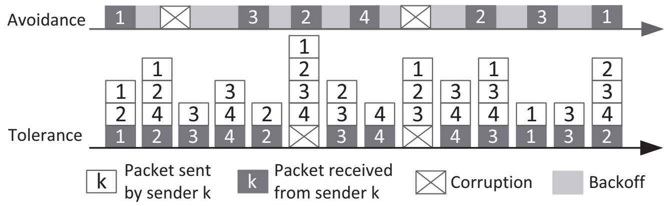


Fig. 1. Channel utilization under collision avoidance and collision tolerance. The ACK packets are omitted for clear display.

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% in general cases, compared with state-of-the-arts protocols.

The rest of this paper is organized as follows. Section II presents in detail the challenges to achieve collision tolerance. In Section III, we present the theoretical model of channel utilization with collision tolerance and analyze the achievable performance gain, compared with collision avoidance. Section IV introduces the design details of $Coco$, followed by the performance evaluation results in Section V. We discuss the related work in Section VI and conclude this paper in Section VII.

II. Challenges in Collision Tolerance

Recent studies on protocol designs pay increasing attention to physical layer (PHY) characteristics, which demonstrate potential opportunities in improving network performance. One of the most promising directions is collision tolerance. In 802.15.4 networks using the DSSS modulation scheme, the tolerance to collisions is manifested as capture effect [3], [4]. Capture effect is a phenomenon that the receiver correctly receives a stronger signal in face of other signals' interference [5]. This phenomenon enables successful transmission of a packet along with collisions. Collision tolerance, however, is a non-trivial 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 direction to overcome those challenges.

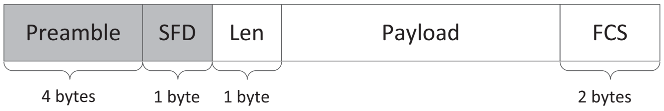


Fig. 2. Format of 802.15.4 packet. In the reception process, preamble is used to detect the beginning of a packet and synchronize the receiver and the sender. After that, SFD triggers the reception of the packet.

We conduct our experiments in ContikiOS [6] with Telos-B [7] motes. There are one receiver and multiple senders in the experiment setting. 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. In the following content, we introduce the challenges from timing and concurrency requirements respectively.

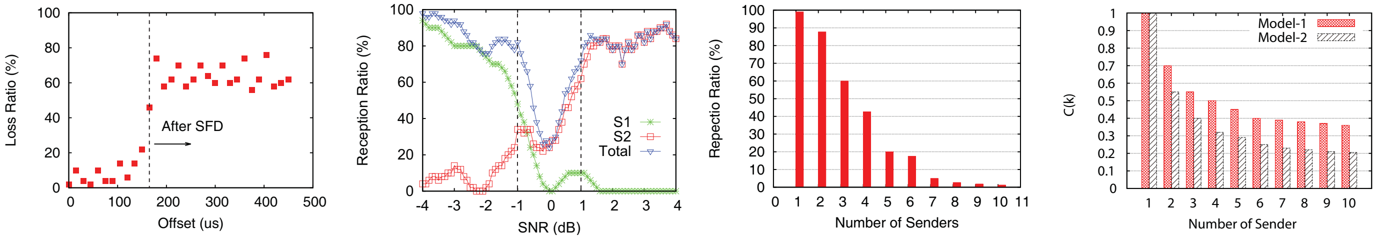
A. Timing Requirement

We first examine the impact of timing, namely the offset of arrival time among the packets. Two nodes are used as senders. One of them transmits a packet once it receives the probe packet from the receiver. The other sender waits for a pre-configured offset time Δt before transmitting its packet. We vary Δt and measure the resulting packet reception ratio at the receiver side. For each Δt, we run the experiment for 100 rounds, in which the senders and receiver are placed at different locations to generate different SNR for each sender.

As we can see from Fig. 3(a), a sharp increase in the loss ratio appears when the offset Δt exceeds 160 μs, which is the time duration of the preamble plus SFD. The reason behind is as follows. When the signals overlap with each other, the receiver always tries to find the strongest one by identifying the preamble and SFD. Therefore when the offset is less than 160 μs, the strong signal is always captured no matter it comes early or late. On the contrary, when the strong signal comes after the SFD of the early-arriving weak signal, it acts as strong interference to the weak one. In this case, the weak signal is corrupted by the strong signal. Nevertheless, the receiver is unable to receive the strong signal because it misses the SFD of the strong signal. Therefore, 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.

B. Concurrency Requirement

For the purpose of collision tolerance, the concurrency (the number of concurrent transmissions) must be limited. Otherwise all packets will be corrupted. We start with an experiment with two senders ($S1$ and $S2$) and one receiver. The senders instantly transmit packets upon receiving the probe from the receiver, so that the above-mentioned timing requirement is satisfied. $S1$ is placed stationarily near the receiver while $S2$ is moved towards the receiver from a far place. At the beginning, $S1$ has a stronger SNR than $S2$. 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 $S1$.



(a) Loss ratio vs. offset of packets

(b) Reception ratio vs. SNR

(c) Reception ratio vs. sender number

(d) $C(k)$ in capture models

Fig. 3. Investigation of difficulties in exploiting collision tolerance. In Fig. 3(a), loss ratio has sharp increase when offset exceeds a value of 160 μs. Fig. 3(b) shows reception ratio for 2-sender case, total reception ratio degrades only when the two senders have nearly strong signal strength. Fig. 3(c) depicts the reception ratio with different number of senders, and reception ratio decreases with the increase of sender number. Fig. 3(d) plots the theoretical $C(k)$ in existing models.

We record the reception ratio at the receiver side and contrast it to the SNR of $S1$ and $S2$. In Fig. 3(b), we can see that with the increase of $S2$'s SNR, the reception ratio of the two senders' packets demonstrate 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., an SNR near 0 dB in Fig. 3(b). For all the cases with two senders, the wireless channel shows high tolerance of collisions.

We increase the number of senders from 2 to 10 to further observe collision tolerance. The senders are randomly placed at different locations near the receiver. For each quantity of senders, we repeat the experiments for 100 rounds. Fig. 3(c) shows that the reception ratio decreases accordingly when total number of senders goes up. 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 4. Note that in this series of experiments, we did not deliberately differ the SNR of the senders. The experimental results apparently indicate the wide applicability of collision tolerance as well as its requirement on transmission concurrency. Compared with other techniques to achieve collision tolerance, e.g. power control [8], controlling the transmission concurrency (i.e., limiting the number of concurrent senders) is relatively effective, cost-efficient, and easy to implement in practice.

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 IV.

III. Theoretical Formulation

A. 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 basic element of time. The duration of a packet transmission generally consists of multiple time slots.

At the receiver side, a time slot should be in one of the 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*, where 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 transmitting while the others stay silent. With collision tolerance, *Coco* increases P_s by exploiting successful packet transmissions while allowing the occurrence of collisions. 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* [9], which means the probability that a packet can be correctly received from collision when k senders are transmitting concurrently. In [9], [10], various models are proposed to measure the capture probability considering the power difference, fading and multipath effect. We follow the model in [10] which is widely adopted:

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

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. Fig. 3(d) plots $C(k)$ with different parameter settings. The result is consistent with the experimental result in Fig. 3(c).

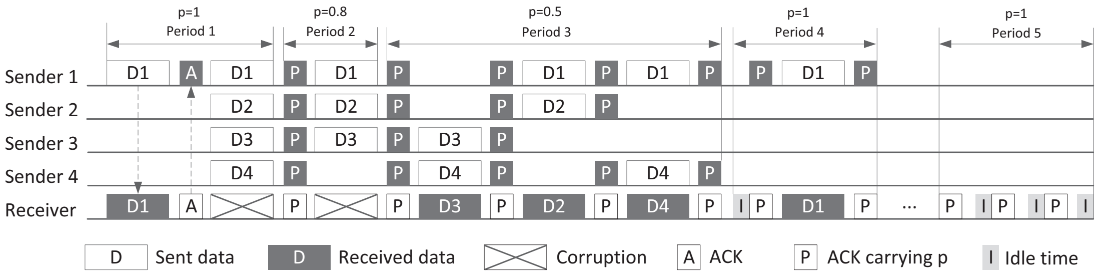


Fig. 4. Overview of *Coco*. Four senders transmit packets to a common receiver. *Coco* starts when collision is detected (Period 1), and probability p is adjusted for best concurrency (Period 2, 3, 4). All senders transmit with the assigned p by the receiver. *Coco* ends when all senders finish their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

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 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 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 to exploit collision tolerance. 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.

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

$$p^{opt} = \arg\max_{p} Util(p) \tag{7}$$

B. Improvement against Collision Avoidance

It is 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. The probability $P_s^{'}$ for random backoff based protocols is:

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

In such protocols, senders take random backoffs and it is therefore difficult to calculate the duration of collisions T_c. Conservatively we use T_s as the upper bound of T_c, similarly with the analysis in [11]. Fig. 5 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% with $\eta = 10$, 10% with $\eta = 30$ and 8% with $\eta = 90$.

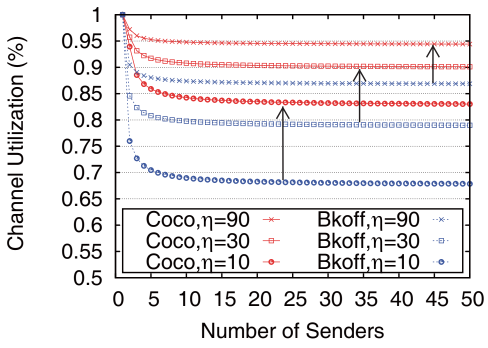


Fig. 5. Comparison between *Coco* and the backoff-based mechanism, with respect to channel utilization. *Coco* improves the achievable upper bound of channel utilization with different $\eta = T_c/T_{slot}$.

Note that the achievable channel utilization for random backoff based protocols seems as high as 86% when $\eta = 90$. In practice, however, it is 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. As a result, the global optimum cannot be reached with an increasing number of senders [12]. The author in [1] reveals the performance degradation with a large number of senders for 802.11 DCF. In comparison, *Coco* is able to 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.

IV. 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 and key feature of *Coco* is to align the packet transmissions

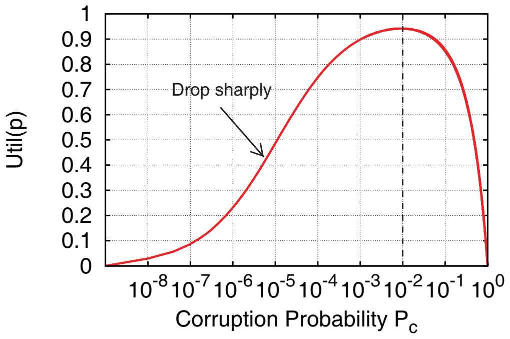


Fig. 6. The impact of P_c on $Util(p)$. On the left side of P_c^{opt}, $Util(p)$ drops quickly from P_c^{opt} to 0 while it changes much more slowly on the right side. (Note the log-scale of X-axis.)

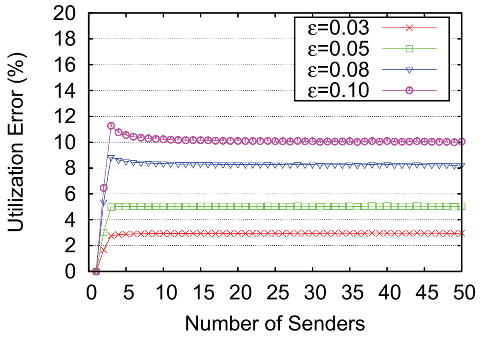


Fig. 7. Error introduced by ϵ for different number of senders.

from multiple concurrent senders and control the transmission probability p of every sender, so that the channel is fully utilized.

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 and discuss its limitations.

A. Overview

Fig. 4 depicts the workflow of $Coco$ with an example of 4 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. Detecting 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 the third subsection) and piggybacks the new value of $p = 0.8$ in the ACK packet.

In period 2, $S1$, $S2$ and $S3$ decide to transmit and suffer 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 an 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 subsections, we elaborate on the two components: (1) The mechanism for precise sender alignment and (2) The online feedback control algorithm to regulate p.

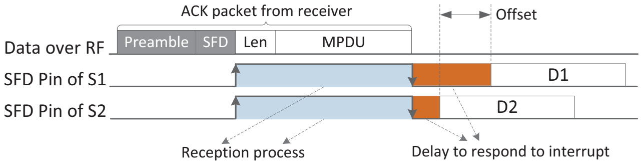


Fig. 8. Senders are aligned by SFD interrupts of ACK packet. Transmission offset are mainly caused by delay in responding the falling-edge interrupt.

B. Align Senders' Packets

The first difficulty mentioned in Sec. 2 can be overcome with the help of ACK mechanism. Instead of transmitting randomly in traditional protocols, senders transmit by detecting ACK packet broadcasted by the intended receiver. The principle is that SFD rising edge and falling edge are strictly aligned for all senders during reception of an identical ACK. Fig. 8 shows two senders are triggered by an ACK, their reception process are synchronized by SFD interrupt. Based on this observation, the offset only occurs when senders respond the interrupt of SFD falling edge. In the evaluation part, we show that this delay is extremely small. In this way, senders with 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 [13], [14]. 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.

C. Feedback Control Algorithm

In Section III, we analyze that p_i^{opt} can be numerically found for different numbers of senders i. However, it is hard in practice, due to the following reasons. First, the receiver has no idea about the total number or the identity of senders. Thus it cannot directly calculate p_i^{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. Some senders may join in the concurrent transmissions

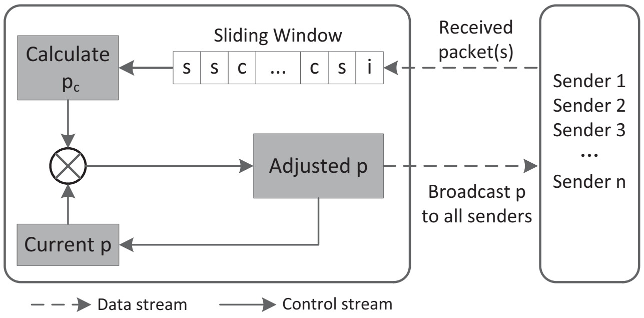


Fig. 9. The control diagram for receiver. Receiver regard the received packet(s) as input and calculate P_c by counting the number of corrupted slots (n_c) in the sliding window with size W. The output is the adjusted probability p, which is carried in ACK and broadcasted to all senders.

TABLE I. p^{opt} AND P_c FOR 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

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_i^{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_i^{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 p_i^{opt}? (3) What is the convergence speed and (4) what are the error bounds?

For the first question, we demonstrate via simulation that P_c^{opt} converges to a constant value quickly, as is shown in Tab. I. We set $\eta = 60$ here for general analysis. 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} does not change when N increases. This property aids us to decide whether p is proper or not. We can first calculate P_c from the statistical number of corrupted slots in a time period and then compare P_c with P_c^{opt}, as is shown in Fig. 9. If P_c is larger than P_c^{opt}, which means that p is too large, we decrease p, otherwise we increase p. Note that when $N = 1$, P_c is always 0, in which case we increase p until 1.

When 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 Alg. 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 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 (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:

Algorithm 1 The Feedback Control Algorithm

```
Input: $n_c$, $W$
Output: $p$
$P_c \Leftarrow n_c/W$                                  # Calculate $P_c$
if $P_c^{opt} \le P_c < P_c^{opt} + \epsilon$ then
    return $p$                                          # p is proper
else if $P_c < P_c^{opt}$ then
    $p_l \Leftarrow p$
    $p \Leftarrow (p_l + p_u)/2$                        # Increase p
    return $p$
else if $P_c \ge P_c^{opt} + \epsilon$ then
    $p_u \Leftarrow p$
    $p \Leftarrow (p_l + p_u)/2$                        # Decrease p
    return $p$
end if
```

$$P_c^{opt} \le P_c < P_c^{opt} + \epsilon \qquad (9)$$

where ϵ is the error bound of P_c. Note that we restrict P_c to a range instead of the exact value P_c^{opt}, so as to avoid fluctuations in feedback control. Besides, the range is set to begin from P_c^{opt} rather than from $P_c^{opt} - \epsilon$. The reason is that the value of $Util(p)$ varies greatly when P_c is in the range from $P_c^{opt} - \epsilon$ to P_c^{opt}. Fig. 6 shows how $Util(p)$ changes on the left and right side of P_c^{opt} (0.0107). It is easy to see that $Util(p)$ decreases to 0 quickly on the left side. Note that X-axis is log-scale.

Error introduced by ϵ. As P_c is not exactly equal to its optimal value P_c^{opt}, the real utilization function $Util(p)$ may not be able to achieve its optimal $Util^{opt}$ as shown in Fig. 5. 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. 7. 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%. Therefore we choose $\epsilon = 0.05$ in our implementation.

Convergence speed. Another performance metric we care is the convergence speed of Alg. 1. We claim that in this algorithm p can converge quickly from its initial value to the approximate value p^{opt}. From Tab. I, we see that p^{opt} 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 for at most $4.096\ ms$. The total convergence time is at most $10 \times 100 \times 4.096\ ms$, which is equal to $4.1\ s$.

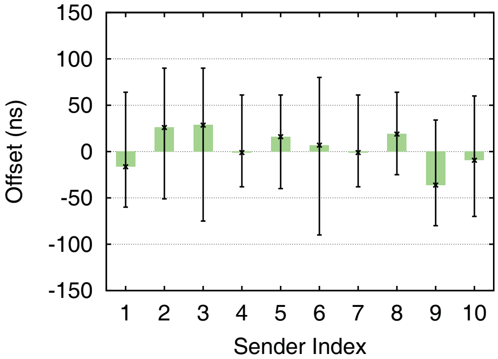


Fig. 10. Transmission offset between senders. Offset is measured between 10 different senders with respect to a reference sender on the receiver side.

D. Discussion

As a MAC layer protocol, *Coco* can be a good candidate substitution of the traditional CSMA backoff mechanism for various protocol scenarios, such as data aggregation [15], relay [16] and data collection. Especially when data rate is high and nodes are dense [17], *Coco* shows it advantages.

However, we claim that there are two issues for *Coco* to be considered. First, because transmission is triggered by the receiver's ACK, *Coco* is like a receiver-initiated protocol [18]. 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.

V. Evaluation

We implement *Coco* in ContikiOS [6] on TelosB [7] platform. Based on the implementation, we conduct experiments to evaluate the performance of *Coco* from different aspects. First, we verify how well *Coco* copes with the difficulties mentioned in Section II to achieve collision tolerance. Namely, we evaluate the timing accuracy and how well p is adjusted in practical networks. Second, we also conduct macrobenchmark experiments to evaluate performance of *Coco*. We compare the channel utilization with conventional protocols based on collision avoidance. Third, we evaluate the robustness of *Coco* with different network settings.

A. Timing Accuracy

We first evaluate the time accuracy of aligning packets with ACK. As discussed in protocol design, senders are synchronized by SFD interrupts. In this experiment, we measure the delay to handle 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. Fig. 10 shows the average offset, the minimum and maximum offset for different nodes in the experiment.

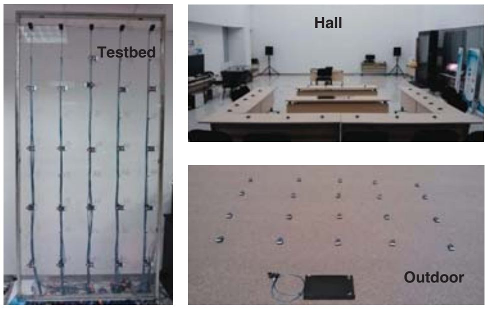


Fig. 11. Environments illustration. A testbed environment is shown in the left figure. We also evaluate *Coco* in an hall and nodes are placed on the desks, which is shown in the top-right figure. The bottom-right figure shows the an outdoor environment and nodes are placed on the ground.

TABLE II. 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

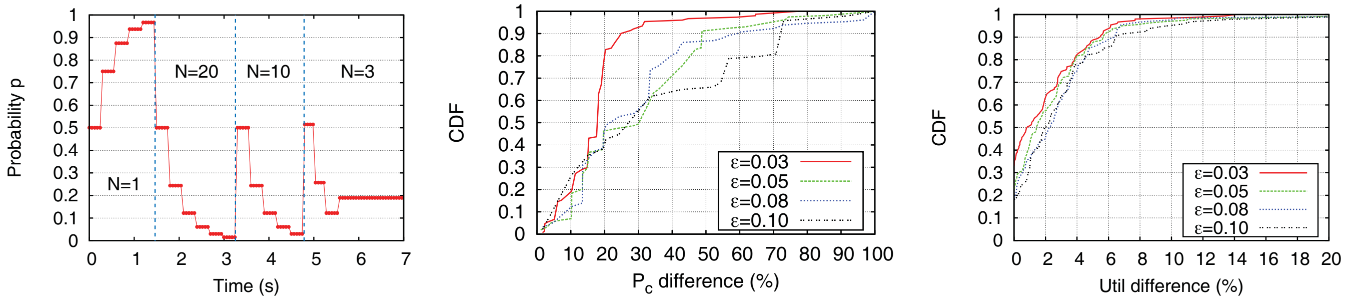
We can see that the maximum offset is about of 100 ns for all 10 senders. this value is far less than the required 160 μs, i.e., the duration of preamble plus SFD byte. Therefore, the timing requirement can be well satisfied by exploiting ACK to align packets on real sensor nodes.

B. Evaluation of Adjustment Algorithm

In this set of experiments, we demonstrate the effectiveness of the feedback control algorithm. In the first experiment, we show the convergence process of p and how p can be adjusted to different number of senders. We manually change the number of senders from 1 to 20, and then to 10, and finally to 3. We record the corresponding value of p during the process. We set the parameter $\epsilon = 0.05$ in this experiment.

Fig. 12(a) 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 adaptive 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 Fig. 12(b) and Fig. 12(c). The impact on P_c is calculated as the ratio of difference between P_c and P_c^{opt} to P_c^{opt}. Error of $Util$ is

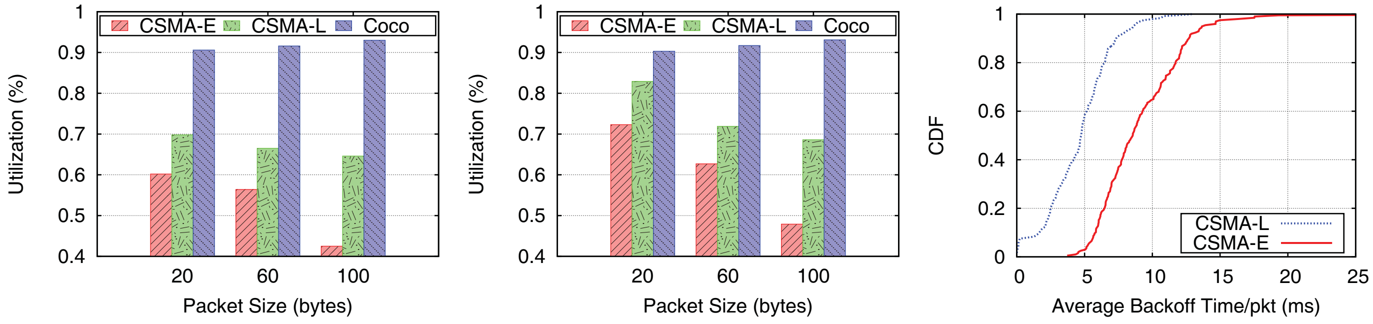


(a) Convergence process of p with $\epsilon = 0.05$.

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

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

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



(a) With Hidden terminal.

(b) Without Hidden Terminal.

(c) Per-packet backoff time.

Fig. 13. Performance comparison with CSMA linear backoff (CSMA-L) and CSMA exponential backoff (CSMA-E) with different packet lengths. Fig. 13(a) shows the results in scenario with hidden terminals and Fig. 13(b) shows the scenario without hidden terminals. Fig. 13(c) shows the CDF of average backoff time per packet in both CSMA-E and CSMA-L.

calculated in a similar way. From Fig. 12(b), we can find that with smaller ϵ, the resulting error of P_c is also small. Fig. 12(c) 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% difference for 80% cases. 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.

C. Protocol Robustness Evaluation

In this subsection, we conduct comprehensive evaluation to the robustness of *Coco* under various network settings. Especially we examine the impact of network environment, network topology and packet size. In all the following three experiments, we use 20 nodes transmitting packets to a common receiver. Each sender transmits 100 packets and we repeat each test for several rounds to calculate statistical channel utilization.

1) Impact of environment: Besides testbed-based experiments, we also conduct experiments in a hall and in an outdoor environment. Fig. 11 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 utilization. Tab. II 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 outdoor environment is slightly better than the other two cases. This experiment result shows that *Coco* can be applied to different environments.

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 Fig. 13(a) and Fig. 13(b). 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 shown in the overall evaluation.

3) Impact of topology: Another parameter that may effect 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 node on the opposite side, resulting 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 strengths.

D. 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 [19] as the representative of random-backoff based protocols.

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 time for all senders to finish transmission at the receiver side to compute the overall utilization.

Fig. 13(a) and Fig. 13(b) show the results. There are several findings to be revealed in this experiment. First, CSMA protocols are not robust to hidden terminal. 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 packet length doesn't affect the corruption probability in *Coco*, while it does for 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 2304 bytes in 802.11 networks. Thus it is desirable to apply larger backoff window in 802.11 networks. 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% to about 50%.

To further investigate the performance of CSMA backoff protocols and *Coco*, we look into the backoff time. Fig. 13(c) 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% of the packets. However, backoff time is not required in *Coco*. Therefore, it is easy to understand the reason for performance gain with *Coco*.

2) Improvement to AMAC: Besides comparison with C-SMA 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 [20], 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, each node polls the channel by sending polling packets to see if there is any transmission to the node. If a node has packets to send, it will sense the channel to see if there is any polling packets from the receiver. If there is, the node will send packets to the receiver.

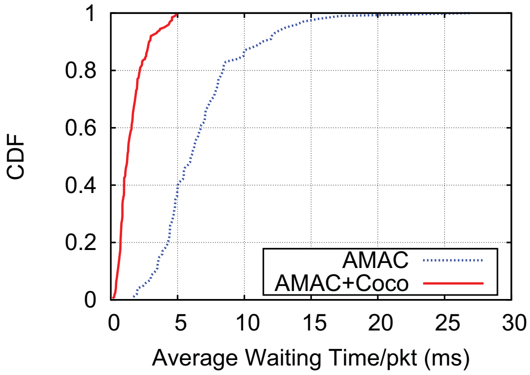


Fig. 14. Per-packet waiting time before and after AMAC is integrated with *Coco*.

As stated in [20], 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 fires, 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. 14, after integrating *Coco* with AMAC, the average used time for each packet is significantly reduced. The average used time per packet is reduced from about 7 ms to only 1 ms in *Coco*. This indicates *Coco* is efficient to resolve collisions in AMAC.

VI. 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 [21]. 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. For the latter, CSMA is applied as the medium access control method and senders perform random backoff for collision avoidance. The overhead in avoiding collision, however, degrades the performance of channel utilization. For example, coordination and synchronization in schedule based protocols consumes 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 [22], Chorus [23] and Glossy [24] 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. 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 by modifying PHY layer. Another category of protocols try to recover collided signals with advanced PHY layer techniques [14], [25], [26], [27]. For example, Zigzag [14] iteratively decodes the collision-free part in the collided signals first and then subtracts it from the collided signal. SIC [25] on the other hand, recovers the strong signal with capture effect and then recover the weak signal by cancelling the strong one from the collided signals. The limitations of these protocols are that they require special modification in the PHY layer and are not supported in commercial hardware. On the contrary, *Coco* is light-weight. Ineeds little modification in the MAC or PHY layer and can be directly used on off-the-shelf hardwares.

VII. 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 [28] 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.

VIII. Acknowledgement

This study is supported in part by NSF China Major Program 61190110, National Basic Research Program (973 program) under Grant of 2014CB347800, NSFC under Grants 61170213, 61202359 and China Post doctoral Science Foundation under grant 2012M520013.

References

[1] G. Bianchi, "Performance analysis of the ieee 802.11 distributed coordination function," *in Procedings of IEEE Journal on Selected Areas in Communications, 2000*, vol. 18, no. 3, pp. 535–547.

[2] K. Wu, H. Tan, H.-L. Ngan, Y. Liu, and L. Ni, "Chip error pattern analysis in ieee 802.15.4," *Mobile Computing, IEEE Transactions on, 2012*, vol. 11, no. 4, pp. 543–552.

[3] K. Leentvaar and J. Flint, "The capture effect in fm receivers," *IEEE Transactions on Communications, 1976*, vol. 24, no. 5, pp. 531–539.

[4] K. Whitehouse, A. Woo, F. Jiang, J. Polastre, and D. Culler, "Exploiting the capture effect for collision detection and recovery," in *Proceedings of IEEE workshop on Embedded Networked Sensors, 2005.*

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

[6] "The contiki operation system," http://www.contiki-os.org/.

[7] J. Polastre, R. Szewczyk, and D. Culler, "Telos: enabling ultra-low power wireless research," in *Proceedings of ACM IPSN, 2005.*

[8] X. Zhang, H. Gong, M. Liu, S. Lu, and J. Wu, "Quantitative analysis of the effect of transmitting power on the capacity of wireless ad hoc networks," in *Proceedings of ACM MobiHoc, 2010.*

[9] M. Zorzi and R. Rao, "Capture and retransmission control in mobile radio," *in Proceedings of IEEE Journal on Selected Areas in Communications, 1994*, vol. 12, no. 8, pp. 1289–1298.

[10] A. Krishna and R. O. LaMaire, "A comparison of radio capture models and their effect on wireless lan protocols," in *Proceedings of IEEE ICUPC, 1994.*

[11] 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 *Proceedings of ACM SIGCOMM, 2005.*

[12] 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 *Proceedings of IEEE INFOCOM, 2012.*

[13] E. Magistretti, O. Gurewitz, and E. W. Knightly, "802.11ec: collision avoidance without control messages," in *Proceedings of ACM MobiCom, 2012.*

[14] S. Gollakota and D. Katabi, "Zigzag decoding: combating hidden terminals in wireless networks," in *Proceedings of ACM SIGCOMM, 2008.*

[15] L. Yu, J. Li, S. Cheng, S. Xiong, and H. Shen, "Secure continuous aggregation in wireless sensor networks," *IEEE Transactions on Parallel and Distributed Systems, 2013*, vol. PP, no. 99, pp. 1–1.

[16] L. Guo, X. Ding, H. Wang, Q. Li, S. Chen, and X. Zhang, "Cooperative relay service in a wireless lan," *in Proceedings of IEEE Journal on Selected Areas in Communications, 2007*, vol. 25, no. 2, pp. 355–368.

[17] K. Jamieson, H. Balakrishnan, and Y. C. Tay, "Sift: A mac protocol for event-driven wireless sensor networks," Tech. Rep., 2003.

[18] 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 *Proceedings ACM SenSys, 2008.*

[19] J. Polastre, J. Hill, and D. Culler, "Versatile low power media access for wireless sensor networks," in *Proceedings of ACM SenSys, 2004.*

[20] 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 *Proceedings of ACM SenSys, 2010.*

[21] 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 Computtin Survey, 2012*, vol. 45, no. 1, pp. 7:1–7:37.

[22] J. Lu and K. Whitehouse, "Flash flooding:exploiting the capture effect for rapid flooding in wireless sensor networks," in *Proceedings of IEEE INFOCOM 2009.*

[23] X. Zhang and K. Shin, "Chorus: Collision resolution for efficient wireless broadcast," in *Proceedings of IEEE INFOCOM 2010.*

[24] F. Ferrari, M. Zimmerling, L. Thiele, and O. Saukh, "Efficient network flooding and time synchronization with glossy," in *Proceedings of ACM IPSN, 2011.*

[25] D. Halperin, T. Anderson, and D. Wetherall, "Taking the sting out of carrier sense: interference cancellation for wireless lans," in *Proceedings of ACM MobiCom, 2008.*

[26] A. Gudipati, S. Pereira, and S. Katti, "Automac: rateless wireless concurrent medium access," in *Proceedings of ACM MobiCom, 2012.*

[27] S. Sen, R. Roy Choudhury, and S. Nelakuditi, "Csma/cn: carrier sense multiple access with collision notification," in *Proceedings of ACM MobiCom, 2010.*

[28] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis, "Collection tree protocol," in *Proceedings of ACM SenSys, 2009.*