Chameleon: On the Energy Efficiency of Exploiting Multiple Frequencies in Wireless Sensor Networks

Jing Li, Wenjie Zeng, and Anish Arora

Department of Computer Science & Engineering
The Ohio State University
{ljing,zengw,anish}@cse.ohio-state.edu

Abstract. We consider the energy efficiency of medium access control (MAC) in low power wireless communication where multiple channels are available and the duty cycle of (send, receive, and idle) channel access is controllable. We show that in this setting maximization of MAC energy efficiency reduces to maximizing the aggregate channel utilization and minimizing the aggregate duty cycle channel access. Based on the reduction, we show the theoretical existence of centralized, global information protocols which achieve optimal energy efficiency in terms of channel assignment and duty cycle scheduling. Then, towards practically realizing these protocols in a distributed fashion with local information only, we present Chameleon, which assigns channels based on lightweight estimation of channel utilization and adapts the duty cycle of node reception relative to the incoming traffic. Chameleon improves energy efficiency and channel utilization not only among users internal to the network, but also in the presence of external users that share the spectrum. We compare Chameleon with state-of-the-art single-channel and multi-channel protocols. Our experimental results show substantial energy efficiency gains over these protocols, which range from an average of 24% to 66%.

Keywords: Energy Efficiency, Multichannel, Duty Cycling, Wireless Sensor Network, TinyOS.

1 Introduction

Energy constraints in wireless sensor networks mandate efficiency of energy spent on communication, sensing, as well as computing. While a good rule of thumb is to design applications whose energy consumption is equal across these three categories, communication energy has dominated in early network deployments. The motivation to particularly improve communication energy efficiency has only increased as the growth in application complexity to date has by far outstripped the growth in available energy.

At the MAC layer, which is a critical component of communication energy efficiency, many protocols have relied on almost-always-off communication. Duty cycling is the norm in state-of-the-art MAC protocols. Ideally the duty cycle should be at a rate that is just sufficient to accommodate the traffic. The choice of the MAC protocol and the duty cycle determine the resulting communication energy efficiency at the MAC layer. In this paper, we consider achievable energy efficiency of duty-cycled MAC operation in networks where multiple channels (equivalently, frequencies) can be exploited.

The few multi-channel protocols that have been proposed in recent years are essentially categorized into four approaches: 1) Statically partition network nodes across multiple channels so that the density of nodes on a given channel is reduced, e.g., MMSN [18] and TMCP [16]; 2) Explicitly negotiate channels to exchange data for collision avoidance based on current usage information of each channel, e.g., MMAC [14] and TMMAC [17]; 3) Migrate network nodes probabilistically at runtime from one channel to another so as to balance traffic load, using control theoretic techniques, e.g., [11] and [10]; and 4) Balance traffic load (deterministically or randomly) across multiple channels evenly so as to reduce potential interference, e.g., Y-MAC [9] for sensor networks and SSCH [3] for more general wireless networks.

All of these approaches significantly improve network goodput and, in turn, energy efficiency, in comparison with MACs that use only one single channel. Several extant protocols do not per se consider duty cycling, but we find that even if one were to include duty cycling along with these approaches, there is room for substantial improvement in goodput and energy efficiency. In the first approach, different channels are assigned to two-hop neighbors to avoid the possibility of interference; since the actual traffic is not considered, it is possible that some channels are lightly loaded and the node partitioning is thus too conservative. This approach also incurs the overhead of distributed distance-2 coloring. For the second approach, although traffic load is considered when assigning frequencies, the explicit channel negotiation for each data communication involves nonnegligible overhead. In addition, the channel usage information has to be updated online within the distance-2 neighborhood. The third approach starts off by utilizing one channel and alleviates unfairness by probabilistically allocating a fraction of nodes into the next channel. In other words, channel utilization is expanded gradually when the goodput drops to a certain empirical threshold as measured in terms of Packet Reception Ratio or percentage of successful channel accesses. Nevertheless the goodput over the available channels is not optimized, nor is the instantaneous condition of every channel taken into account when nodes perform channel switching. As for the fourth approach, although splitting traffic loads evenly over multiple channels achieves fairness, the aggregate goodput of the network is again not necessarily maximized.

None of these approaches choose channels based on a comprehensive (albeit local) view of the current condition of all channels. Thus, the channels to which nodes are switched into may not represent the best choice. This is especially true if we take into account interference that results from the concurrent operation of external networks. Selecting channels based on a locally comprehensive yet lightweight estimator of channel utilization efficiently is the starting point for our design of a multi-channel MAC protocol, Chameleon [1].

Chameleon has two main components for maximizing energy efficiency. One, its multi-channel scheduler, uses the lightweight estimator to select channels in accordance with an optimality analysis presented in the following section. And second, its radio scheduler, uses a receiver centric approach to coordinate senders and receivers with approximately optimal efficiency; receiver centric MACs were independently introduced in OMAC [5] and Crankshaft [7] and shown analytically to have higher energy efficiency than sender-centric MACs [5]; more recently, the RI-MAC receiver centric protocol was experimentally shown to have energy gains over state-of-the-art sender-centric MACs [13,4]. This component also realizes locally adaptive duty cycling, which staggers data communication periods so that the resulting energy efficiency is highest at the chosen duty cycle.

The main contributions of the paper are as follows.

- We formalize the optimization of MAC energy efficiency in a setting where duty cycling and multiple channel utilization is possible. We show that the optimization reduces to maximizing the spectrum utilization over all available channels while minimizing the duty cycle.
- We provide a protocol that optimizes MAC energy efficiency, assuming the existence of two components, one for precisely quantifying node utilization on each channel and the other for minimizing the send-receive-idle duty cycle for a given node traffic.
- We give lightweight implementations that approximately satisfy these two components, and thus obtain the Chameleon protocol that approximates the optimal protocol. Our implementation of the first component uses a lightweight metric w which is passively computed at each receiver node. Our implementation of the second component uses a receiver centric pseudo-random scheduling of wakeup times, so that receivers within each other interference range are unlikely to be up simultaneously; it also chooses the receiver duty cycle to be just enough such that the receiver experiences low sender collision rate. A side-effect of this approach is that Chameleon intrinsically accommodates external interference.
- We validate, using experiments on the TelosB mote platform, that Chameleon is capable of maintaining substantially higher energy efficiency than both representative single-channel and multi-channel MAC protocols, including MMSN, Y-MAC, BoX-MAC, and O-MAC.

The rest of this paper is organized as follows. We present, in Section 2, the system model as well as an analysis of energy efficiency optimization. We discuss

[1] Recent research shows that chameleons change color not to camouflage themselves but to communicate. Their "bandwidth" of communication (aka signalling) is related to the number of colors that they use. Cf.: D. Stuart-Fox and A. Moussalli, "Selection for social signalling drives the evolution of chameleon colour change", *PLoS Biol* 6(1): e25, 2008.

a solution approach for implementing an optimal protocol and design our multi-channel protocol, Chameleon, in Section 3. In Section 4, we present experimental evaluations of relevant aspects. We discuss related work in Section 5 and our conclusions in Section 6.

2 Energy Efficiency Analysis

In this section, we first define channel utilization, spectrum utilization, and energy efficiency. We then discuss maximization of energy efficiency for a receiver given network traffic load, in terms of expected spectrum utilization and duty cycling.

2.1 System Model

The network consists of N energy-constrained half-duplex wireless sensor nodes. Radio operation of each node is represented by a contiguous sequence of frames. Each frame consists of a number of time slots; for ease of exposition, we let this number be a global constant. We define a node's *duty cycle*, implicitly over some number of frames, to be the percentage of the time slots, ψ, when its radio is active; $\psi \in [0, 1]$. A node's duty cycle is further decomposed into its transmit duty cycle, the percentage of the slots when its radio is transmitting, and its receive duty cycle, the percentage of the slots when its radio is in receive or listen mode.

For a given node i, we refer to the packets that are sent to i as its "in-traffic", while packets that are not sent to i but are overheard by i or whose collision is overheard by i are its "interference traffic".

The cumulative wireless bandwidth that can be utilized by nodes denotes the network "spectrum". Spectrum is divided into several orthogonal "channels" (or "frequencies") such that communications on different channels either never or only barely interfere with each other (in practice, adjacent channels are typically not completely interference free from each other [2]). Within each channel, collisions may occur if wireless devices attempt to transmit simultaneously.

The wireless network is viewed as formed by overlapping broadcast domains. Accordingly, we define a receiver's interference set as the set of nodes whose broadcast domain covers the receiver.[2] We let η denote the average size of the interference set for a given node. Let i, j, h range over nodes in the network and k range over channels of the spectrum.

With respect to a given receiver and its interference set, we define the **channel utilization** for a given channel, k, as the ratio, $E(k)$, of the number of time slots where a packet is successfully received to the total number of time slots. (The definition may be relativized to the number of frames considered in the definition of duty cycle.)

[2] We note that several of our definitions are receiver-centric rather than sender-centric, as this significantly simplifies our exposition.

Consequently, **spectrum utilization** with respect to a receiver and its interference set denotes the overall successful transmissions among all channels over the total number of time slots normalized by the number of channels, M. Hence, spectrum utilization is defined as:

$$E_S = \frac{\sum_{k=1}^{M} E(k)}{M}. \tag{1}$$

Our primary interest is in the metric of **energy efficiency**, which refers to the goodput for a given energy budget [5]. Basically, this metric refers to the ratio of the number of time slots with successful receptions to the number of slots in which radios are active, albeit they are transmitting, idle, or active. Eq. (2) defines energy efficiency for a unicast scenario. Notation T in the formula is the total number of slots considered. Compared to channel and spectrum utilization, duty cycling of a node is taken into consideration in the metric.

$$E_E = \frac{\sum_{l=1}^{T} \sum_{j=1}^{N} Z_j^l}{\sum_{l=1}^{T} \sum_{j=1}^{N} (S_j^l + R_j^l)} \tag{2}$$

where

$$S_j^l = \begin{cases} 1, \text{when node } j \text{ transmits in slot } l \\ 0, \text{when node } j \text{ sleeps in slot } l \end{cases}$$

$$R_j^l = \begin{cases} 1, \text{when node } j \text{ listens in slot } l \\ 0, \text{when node } j \text{ sleeps in slot } l \end{cases}$$

$$Z_j^l = \begin{cases} 2, \text{node } j \text{ succeeds receiving its packets in slot } l \\ 0, \text{otherwise} \end{cases}$$

In the following analysis, we focus on exploring how to maximize energy efficiency at the receiver for the case of unicast traffic.

2.2 Energy Efficiency Optimization

Problem Statement. Given a node i, whose interference set is of size η, our goal is to schedule its in-traffic—i.e., choose channels and wakeup times for the i and nodes sending packets to i— such that the resulting energy efficiency E_E of i is maximized.

We approach this problem by first simplifying Eq. (2) for the given node i. First, spectrum utilization reflects the goodput resulting from communications of the nodes in the interference range of node i, which is $\sum\sum Z_j^l$. It follows that $\sum\sum Z_j^l = 2TME_S$, where $2ME_S$ equals the aggregate spectrum utilization and the factor of 2 reflects the benefit to both parties in a communication. The energy consumption of node i, which is determined by the duty cycle control scheme, is $\sum\sum(S_j^l + R_j^l) = T\sum_{j=1}^{\eta} \psi_j$. Thus, the formula below is an equivalent representation of energy efficiency.

$$E_E = \frac{2ME_S}{\sum_{j=1}^{\eta} \psi_j} \tag{3}$$

In order to optimize E_E via maximizing E_S as well as minimizing $\sum_{j=1}^{\eta} \psi_j$, the scheduler has to choose channels and wakeup times. We will first consider channel selection that maximizes the expected spectrum utilization $\hat{E_S}$, then we will discuss how to schedule the wakeup times of nodes to minimize $\sum_{j=1}^{\eta} \psi_j$.

Maximizing $\hat{\mathbf{E_S}}$. Recall that $E(k)$ is the successful reception probability in the interference set of the given node. For the purpose of analysis, in this subsection, we make two assumptions. One is that the in-traffic of nodes follows a stationary process with uniform distribution of arrival times; let the in-traffic load at node i, denoted by p_i, be the probability that on average a packet is sent to i. And two, that the node and its interference set form a clique, i.e., each of these nodes can overhear each packet sent by another of these nodes; thus if packets are concurrently sent to different nodes, collisions will result at each receiver. It follows that all nodes in the network hold the same $E(k)$, which is defined by Eq. (4).

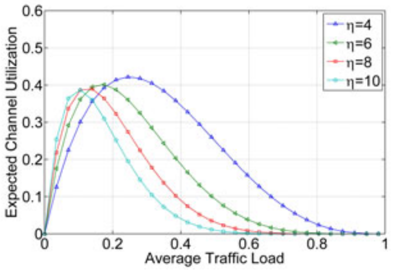
$$E(k) = \sum_j p_j \prod_{h \neq j} (1 - p_h) \tag{4}$$

where j and h range over these nodes. Initially, $E(k)$ increases as traffic loads increase. However, utilization decreases when the channel becomes overloaded, in which case collisions (or, in a contention based scheme, backoff procedures) dominate the communication.

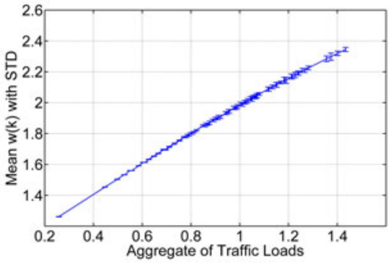
Lemma 1. *The expected channel utilization with respect to node i, $\hat{E}(k)$, is maximized when the aggregate traffic load in the interference set of i, $\sum_{j=1}^{\eta} p_j(k)$, increases to 1.*

Proof. The average traffic load on channel k is $\bar{p}(k) = \sum_{j=1}^{\eta} p_j(k)/\eta$. Hence, by Eq. (4), the expected channel utilization $\hat{E}(k) = \eta\bar{p}(k)(1 - \bar{p}(k))^{\eta-1}$. Fig. 1(a) plots how $\hat{E}(k)$ changes as $\bar{p}(k)$ changes with interference size η. The expected channel utilization is maximized when $\bar{p}(k) = 1/\eta$. Since $\bar{p}(k) = 1/\eta$ implies $\sum_{j=1}^{\eta} p_j(k) = 1$, it follows that maximal utilization is achieved when the aggregate load, $\sum_{j=1}^{\eta} p_j(k)$, equals 1. Alternatively speaking, $\hat{E}(k)$ increases as the aggregate load increases up to 1; after reaching 1, $\hat{E}(k)$ decreases as the aggregate load increases. Hence, the total traffic load should be 1 to achieve maximal channel utilization $\hat{E}(k)$.

Lemma 1 corroborates two facts: 1) the aggregate traffic load, $\sum_{j=1}^{\eta} p_j(k)$, is a judicious estimator of the expected channel utilization; and 2) when the estimator equals 1, channel utilization is expected to be optimum.



(a)



(b)

Fig. 1. (a) $\hat{E}(k)$ vs $\bar{p}(k)$ and (b)Mean $w(k)$ with std vs Metric $\sum p_j(k)$

Now, let us consider channel selection for in-traffic p_i at receiver i. Theorem 1 states a sufficient condition for selecting channels for load p_i that maximize $\hat{E_S}$.

Let $\bar{p_I}(k)$ be the average interference load over η on each channel k. Define $q(k)$ as 1 minus the current total load on channel k, i.e., $q(k) = 1 - \eta\bar{p_I}(k)$. Let $\boldsymbol{q}$ be the vector of qs for all channels that is sorted in a nonincreasing order. Thus, q_s represents the s-th greatest element in $\boldsymbol{q}$, corresponding to channel of index $C(q_s)$. Let vector $\boldsymbol{\alpha} = \{\alpha(k) : k = 1, ..., M\}$ denote the percentages of in-traffic allocated to each channel, i.e., $\alpha(k) \cdot p_i$ is loaded on channel k.

Theorem 1. *$\hat{E_S}$ is optimized if we allocate traffic load p_i to channels according to fractions $\boldsymbol{\alpha}$ computed in Eq. (5).*

$$\alpha(C(q_s)) = \begin{cases} \frac{q_s}{p_i}, & p_i - \sum_{t=1}^{s-1} q_t \geq q_s \\ \frac{p_i - \sum_{t=1}^{s-1} q_t}{p_i}, & 0 < p_i - \sum_{t=1}^{s-1} q_t < q_s \\ 0, & p_i - \sum_{t=1}^{s-1} q_t \leq 0 \end{cases} \tag{5}$$

Proof. $\boldsymbol{q}$ represents residual quota of load on each channel for maximizing channel utilization according to Lemma 1. The essential idea here is to prioritize filling up channels based on $\boldsymbol{q}$, i.e., giving preference to those which have more residual capacity, until load p_i has been assigned completely or all qs in $\boldsymbol{q}$ have been consumed.

Define Δp to be the smallest unit of load that can be assigned on a channel. Hence, load p_i consists of $\lceil p_i/\Delta p \rceil$ units. Before adding a unit Δp into channel k, the expected utilization on channel k is $\hat{E}(k) = \eta\bar{p_I}(k)(1 - \bar{p_I}(k))^{\eta-1}$. After adding Δp, by Eq. (4), the expected utilization becomes $\hat{E}'(k) = \eta\bar{p_I}(k)(1 - \Delta p)(1 - \bar{p_I}(k))^{\eta-1} + \Delta p(1 - \bar{p_I}(k))^{\eta}$. Thus, the utilization gain, $\Delta\hat{E}(k)$, on channel k after appending each Δp would be

$$\Delta\hat{E}(k) = \Delta p(1 - (\eta+1)\bar{p_I}(k))(1 - \bar{p_I}(k))^{\eta-1}, \tag{6}$$

which is a monotone decreasing function of $\bar{p_I}(k)$. The smaller the $\bar{p_I}(k)$, the higher the utilization gain will be. Since Δp is an atomic unit, assigning the channel with lowest $\bar{p_I}(k)$ will provide the highest $\Delta\hat{E_S}$, where $\Delta\hat{E_S} = \Delta\hat{E}(k)$ and k is the channel assigned to the Δp load.

Ideally, all $\lceil p_i/\Delta p \rceil$ units would be added into the channel with the lowest $\bar{p_I}(k)$ to maximize total utilization gain. According to Lemma 1, however, the total load on each channel k should not exceed 1 to achieve maximal utilization. $C(q_s)$ denotes the channel which has s-th smallest $\bar{p_I}(k)$ and $q_s/\Delta p$ is the number of units that can be added to a given channel before exceeding the maximum. Therefore, sequentially filling up each channel according to the order in $\boldsymbol{q}$ will maximize total $\hat{E_S}$.

Consider the assignment of load to channel $C(q_s)$. The number of load units of p_i that are yet to be assigned is $(p_i - \sum_{t=1}^{s-1} q_t)/\Delta p$. If this number is non-positive, indicating that all units of p_i have been assigned to channels earlier in the order of $\boldsymbol{q}$, the fraction assigned to channel $\alpha(C(q_s))$ is 0. Otherwise, if the number of unassigned load units is less than q_s, we can assign all of $(p_i - \sum_{t=1}^{s-1} q_t)/\Delta p$ units to channel $C(q_s)$. $\alpha(C(q_s)) = (p_i - \sum_{t=1}^{s-1} q_t)/p_i$ in this case. If the number of unassigned units is not less than q_s, we can fill up this channel with $\alpha(C(q_s)) = q_s/p_i$.

In essence, Theorem 1 yields one approach for optimizing the spectrum utilization by choosing channels for the in-traffic at a node.

Minimizing ψ. We now consider scheduling for duty cycle minimization. It is straightforward to show a "centralized TDMA and duty cycling" scheduler that has full information of the arrival times of all packets would suffice to this end. This scheduler (having scheduled the existing traffic in the network) can schedule packet communication time so that no collisions happen, as well as senders and receivers are scheduled to wakeup exactly at these times. Lemma 2 states that nodes running the duty-cycled TDMA will minimize gross duty cycle $\sum_{j=1}^{\eta} \psi_j$.

Lemma 2. *Given traffic load p_i and the arrival time of the in-traffic of i, the centralized TDMA and duty cycling scheduler minimizes the total duty cycle $\sum_{j=1}^{\eta} \psi_j$.*

Proof. Duty cycles of nodes that are neither senders nor receivers of packets in the in-traffic of i will remain unchanged. As for nodes involved in the traffic, the scheduler trivially minimizes the wakeup times, since there are no superfluous sends or receives or idle slots. The total duty cycle consumed by the load p_i is minimized to be twice of the load, i.e., $2p_i$.

3 Energy Efficient Multi-channel Protocol Design

In this section, we present our energy efficient multi-channel access protocol, Chameleon. First, guided by Theorem 1 and Lemma 1, we present the component that (re)assigns channels to load units. Then, we design a light-weight, local, receiver-centric scheduler that approximates the heavy-weight centralized scheduler indicated in Lemma 2. We conclude with an overview of our TinyOS implementation of Chameleon.

3.1 Channel to Load Assignment

In Lemma 1, channel utilization is estimated via the sum of traffic loads, including in-traffic load and interference load. These two loads also determine channel assignment according to Theorem 1. Basically, each node, say i, continually performs three tasks: (i) determines the in-traffic for i, (ii) determines the interference traffic to i, and (iii) chooses channels for the in-traffic according to Eq. (5).

For task (i), the in-traffic load at node i (i.e., the exact instantaneous p_i value) can be computed either by appending rate information to data packets sent to receiver i or by locally calculating the rate of incoming load at i; we chose the former.

Task (ii) involves collecting information about the interference load at node i, $\eta \bar{p_I}(k)$ for each channel k. Rather than let i actively coordinate with all nodes in its interference set to compute the value, we introduce a local interference estimator for $\eta \bar{p_I}(k)$ in the next subsection.

Local Interference Estimator. Let interference estimator $I(k)$, defined in Eq. (7), refer to the probability that some interferers of node i transmit on channel k.

$$I(k) = 1 - (1 - \bar{p_I}(k))^{\eta} \tag{7}$$

It follows that $\eta \bar{p_I}(k)$ is estimated by the exponential function of $I(k)$, i.e., $e^{I(k)}$. We leverage the similarity between the sum of traffic loads, notated by $\sum_{j=1}^{\eta} p_j(k)$, and $p_i(k) + e^{I(k)}$, denoted by $w(k)$. Hence, $w(k)$ is employed to compute channel utilization.

Fig. 1(b) shows an instance of the relation between $\sum p_j(k)$ and $w(k)$. We consider a clique network wherein six pairs of nodes communicate independently on the same channel, each with an arbitrarily chosen traffic load in the range [0,1]. Each receiver locally computes the metric $w(k)$, where k is fixed. Fig. 1(b) plots the mean value $w(k)$ and the standard deviation of the six receivers versus the aggregate traffic load $\sum p_j(k)$. Here, the same value of the aggregate load corresponds to a few different sequences of traffic loads $\boldsymbol{p}$. We observe in the figure that $w(k)$ is approximately linear with $\sum p_j(k)$, which verifies that $e^{I(k)}$ is a feasible estimator for interference load, in lieu of the metric $\eta \bar{p_I}(k)$. Additionally, the locally computed deviation of the $w(k)$s is very small, i.e., the average standard deviation shown in the figure is around 0.005. Another relevant observation from our analysis is that when $\sum p_j(k)$ equals 1, $w(k)$ is equal to 2 (see the figure). This is the state where $\hat{E}(k)$ is optimized, and we refer to it as w^*. Moreover, when parameter $\eta > 2$ and $e^{I(k)} \leq w^*$, the linear relation between

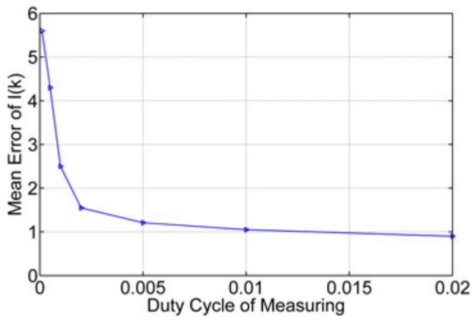


Fig. 2. Mean Error of Measured I(k) versus the Duty Cycle of Measurement

$\sum p_j(k)$ and $w(k)$ is preserved for different configurations of η. It follows that metric $q(k)$ in Theorem 1 may be substituted by the local metric $w^* - e^{I(k)}$ as we perform task (iii).

In particular, the computation of $I(k)$ does not involve sending any specific information, in contrast with the Channel Access Ratio message used in many multi-channel protocols, such as [11]. We explain how interference level $I(k)$ is measured in the next subsection, and how the local metric is used in task (iii) in the following subsection.

Estimator Implementation. The value of $I(k)$ is measured passively at node i by randomly listening to channel k when i is not performing data reception or data transmission. Measurement is performed periodically (at a low duty cycle). For each period, the ratio of the number of busy slots to the total number of checked slots yields the value for $I(k)$.

In terms of implementation, we let nodes perform a continuous Clear Channel Assessment (CCA) check on a given channel during each check slot to determine whether that slot has interference traffic or not. (For the TelosB platform, we empirically chose the channel monitor slot length to be 3ms.) Due to the inefficiency of float operation in the mainstream sensor platforms, we normalize and quantize load into integer "levels". We let the unit of load, δp, be 0.01; 0.01 thus corresponds to the integer level 1. Traffic p_i and interference $I(k)$ are normalized to $\lceil p_i/\delta p \rceil$ and $\lceil I(k)/\delta p \rceil$, respectively. Furthermore, we pre-compute the corresponding value of $e^{I(k)}$ for each level of $I(k)$, thus every receiver maintains a vector $\mathbf{exp}\,\boldsymbol{I} = \{e^{I(k)} : k = 1, 2, ..., M\}$, representing the interference traffic load on each channel.

The choice of measurement period involves a tradeoff between accuracy and energy consumption. To understand this tradeoff, we conducted experiments in which all 5 nodes transmit independently at a specified rate. Each experiment was repeated for traffic loads of 0.01 (approximately 1 packet per second), 0.05, and 0.1, respectively, and also with the nodes performing channel measurement at different duty cycles. We let channel monitoring be triggered by a randomized timer that fires between $0.5T$ and $1.5T$, where $T = 15s$. When the timer fires, the node monitors the channel for several slots if the radio is not being occupied; otherwise, it waits to measure until the radio is released by other processes. The cumulatively measured value of $I(k)$ is reported at a fixed interval of every 5 minutes. To further reduce error, a weighted moving average to consecutive measurements is computed. Hence, $I(k) = \alpha I(k) + (1 - \alpha)I'(k)$, where $I'(k)$ is the value of last measurement. We let α be 0.6 in our experiments. After each report, the counter of $I(k)$ is cleared to zero and another period of monitoring started.

Fig. 2 plots the mean error between the measured level and the expected value with the monitoring channel at different duty cycles ranging from 0.01% to 2%. The x axis represents the duty cycle of passive channel monitoring. Initially, as the monitoring duty cycle increases, the precision of measure increases significantly; however, the improvement reduces when duty cycle is greater than 0.2%. The corresponding average error is at a level of 1 to 2. Thus, to update channel interference level $I(k)$ at an interval of 5 minutes, a duty cycle of 0.1% to 0.2% for channel measurement seems adequate. Alternatively, checking randomly every 200 slots would provide an acceptable measurement for a channel (recall that each slot is 3 ms).

Chameleon offers upper layers the option to adapt channel update interval from time to time to deal with dynamic environments. In the following experiments, we use a 0.2% duty cycle for interference monitoring, unless stated otherwise.

Algorithm for Channel to Load Assignment. Having obtained in-traffic and interference load, task (iii) is implemented by Algorithm 1. Given normalized levels of p_i and $\mathbf{exp}\,\boldsymbol{I}$, we first compute the number of acceptable units on each channel, in $\boldsymbol{q}$ (lines 1 to 7). Lines 17 to 26 assign units to each channel according to Theorem 1, which results in a vector $\boldsymbol{V}$ of size M, e.g., $\boldsymbol{V} = (3, 7, 1, 0, ..., 0)$, where each element represents the units allocated to the channel. Thus, p_i is split across the channels in proportion to $\boldsymbol{V}$. (Which channels to use in which frame is discussed later in this section.) If the sum of the available capacity, $\sum_{k=1}^{M} q(k)$, is less than the total p_i, cf. line 9, senders are notified to reduce their outgoing traffic if possible.

Each node starts with conducting a cumulative measurement for every available channel, followed by independently allocating its load to the corresponding channels. As network load varies, the channel monitoring daemon updates channel assignment (in vector $\boldsymbol{V}$) at each receiver. To alleviate fluctuations caused by simultaneously channel switching, every receiver carries out its channel reassignment with a random interval.

3.2 Receiver-Centric Wakeup and Channel Scheduler

Lemma 2 indicates that there exists in theory a centralized, global information scheduler for maximizing energy efficiency. The scheduler continually performs for each node, say i, the following task: it computes the time at which each in-packet at i is sent without interfering with any of the packets scheduled thus

Algorithm 1. *Channel to Load Assignment*

```
Require: $p_i, \exp \boldsymbol{I}$
 1: for $k = 1$ to $M$ do
 2:     if $w^* - e^{I(k)} \leq 0$ then
 3:         $q(k) \leftarrow 0$;
 4:     else
 5:         $q(k) \leftarrow w^* - e^{I(k)}$;
 6:     end if
 7: end for
 8:
 9: if $\sum_{k=1}^{M} q(k) < p_i$ then
10:     Inform senders (optional);
11: end if
12:
13:
14: Sort $\boldsymbol{q}$ in non–increasing order
15: $\boldsymbol{q} = (q_1, q_2, ..., q_M)$
16: the channel index of $q_s$ is $C(q_s)$;
17: for $s = 1$ to $M$ do
18:     if $p_i - \sum_{t=1}^{s-1} q_t \geq q_s$ then $V(C(q_s)) \leftarrow q_s$;
19:     else
20:         if $p_i - \sum_{t=1}^{s-1} q_t < q_s$ then
21:             $V(C(q_s)) \leftarrow p_i - \sum_{t=1}^{s-1} q_t$;
22:         else
23:             $V(C(q_s)) \leftarrow 0$;
24:         end if
25:     end if
26: end for
```

far; it also updates the sleep-wakeup schedule of the nodes so that they wake up only when they are involved in transmitting or receiving each in-packet to i. Note that the packet transmission time scheduling yields an in-traffic whose arrival time may no longer satisfy a uniform distribution, which we assumed in the analysis shown in Section 2, but since this scheduler enforces collision freedom, the expected E_S and E_E are not negatively affected. However, this centralized scheduler is of high complexity.

Wakeup Scheduler. We now discuss a distributed, light-weight component that efficiently approximates the centralized scheduler. Specifically, we adopt a synchronous, receiver-centric scheduling approach that locally avoid collision and schedules sleep-wakeup. This approach is exemplified by O-MAC [5] and Crankshaft [7]; the approach is in contrast to RI-MAC [15], which is also receiver-centric but is asynchronous.

The basic idea that we borrow from synchronous receiver-centric MACs is this: Each receiver has a pseudo-random scheduler which determines its wakeup slots. The wakeup schedule is advertised to neighbors, compactly since essentially the pseudo-random seed needs to be shared, via a neighbor discovery process. When a node discovers this receiver, it also obtains this receiver's state (of pseudorandom generation), and thus the node can generate the receiver's wakeup schedule. When the node wishes to send to the receiver, it wakes up at the next slot at which the receiver will be awake and attempts to communicate. Two basic modules, neighbor discovery and time synchronization, are used and in turn the module offers Send and Receive interfaces.

Chameleon adopts these basic interfaces from those in O-MAC. This decentralized pseudo-random scheduling staggers nodes' wakeup times with high probability, and has been proven [5] to utilize less duty cycle (i.e., to have higher energy efficiency) under the same traffic load in network than other sender-centric protocols, such as B-MAC, BoX-MAC [13], X-MAC [4] and others. As

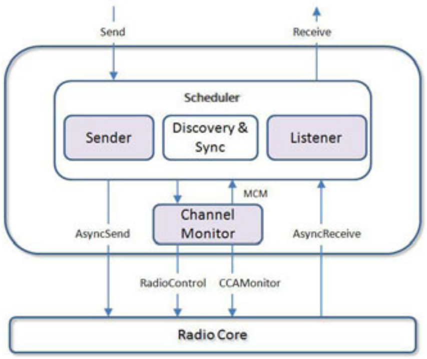


Fig. 3. Composition of the Chameleon Protocol

compared to asynchronous receiver-centric protocols, such as RI-MAC, a sender in O-MAC will not wakeup for an average of half a frame waiting for beacon from its receiver, thus the duty cycle at sender side of O-MAC is obviously less than that of RI-MAC although the receiver's duty cycles are comparable in two protocols.

In other words, in Chameleon's receiver-centric scheme, the senders' wakeup times are implicitly scheduled. Since receivers wakeup at random times in each frame, the likelihood that two interfering receivers will simultaneously receive is low. In O-MAC, a short beacon is broadcast by the receiver as it wakes up to compensate for slot misalignment with potential senders. The beacon contains an adaptive contention window size determined by the receiver side for collision avoidance, based on the expected number of concurrent senders for that receiver. O-MAC is also flexible in adapting duty cycle to incoming traffic load. A sender is allowed to continuously send queued packets to a receiver as long as the sender grabs the channel for the first packet. When a node fails in competition, it continues to compete for the next frame.

Channel Scheduler. We extend the basic O-MAC scheduler in two ways: 1) channel association with frames; 2) channel notification from receivers to senders.

First, the scheduler associates a channel with each frame. This channel is used by the receiver in all slots in which it wakesup during that frame. We implement this association using a vector of units assigned to each channel, $\boldsymbol{V}$, which has size M. Given an assignment $\boldsymbol{V}$, the receiver maintains a shadow copy $\boldsymbol{V'}$, which is initially set to $\boldsymbol{V}$. In each frame, it checks the next k in $\boldsymbol{V'}$. If the value of $V(k) > 0$, then channel k is used in the next frame and the current value in $\boldsymbol{V'}$ is decremented; otherwise, the next channel is checked until all values become 0. Then, $\boldsymbol{V}$ is copied to $\boldsymbol{V'}$ again and the above procedure repeated. In this way, nodes uses multiple channels in proportion to $\boldsymbol{V}$. Equivalently speaking, the incoming traffic to the node is split over multiple channels.

Second, there are two ways in which the receiver shares its updated channel assignment with senders in the receiver-centric approach: asynchronously, through the neighbor discovery process and, synchronously, through beaconing in the first wakeup slot at the beginning of each frame. In the former case, nodes independently compute each other's wakeup slot and channel. The updated channel-wakeup schedule V has to be notified to neighbors via the discovery module within certain amount of time. Each sender keeps its own updated $\boldsymbol{V'}$ and the current index of the receiver, generating future wakeup slots and channels independently. This scheme is realized by leveraging the asynchronous neighbor discovery protocol, Disco [6], which schedules radio wake times at multiples of prime numbers, ensuring deterministic pairwise discovery and rendezvous latency. Disco operates on a wellknown channel, called the home channel. We add several small pieces of information to the packets sent out by Disco, related to time synchronization, channel assignment, and wakeup schedule. When a receiver starts to change its channel assignment schedule, it may accelerate propagating a channel update, by increasing the duty cycle of Disco. After exceeding the deterministic rendezvous period, Disco goes back to previous low duty cycle. The energy cost of updating schedules through Disco is nontrivial, especially when frequent updates exist. Moreover, the discovery schedule may interrupt with node's listen schedule more frequently in this case.

In the latter case, status is updated by advertising the receiver's current channel at the beginning of each frame, using the home channel. Senders do not maintain any channel information, instead they listen to the home channel during the wakeup slot of receiver. The receiver broadcasts the channel it is going to use for current frame in a short beacon on the home channel. Note that the beaconing is part of O-MAC protocol. Following this beacon, potential senders and receiver all switch to the chosen channel for the rest of communication. Specifically, receiver switches to the chosen channel after sending out beacon and senders change to the channel after receiving the beacon. The total beaconing and channel switch time is approximately 5 ms on the TelosB platform.

3.3 Implementation

We implemented Chameleon in TinyOS 2.x for the CC2420 radio platform, which is a packetizing radio used in popular TelosB and MicaZ motes; the code is readily ported to motes with streaming radios such as the CC1000. The composition of Chameleon is shown in Fig. 3.

The Scheduler module in Fig. 3 includes three basic modules provided by O-MAC: listener, sender, and discovery & synchronization. The Listener module decides node wakeup times and durations, while Sender determines when to transmit application packets given the state maintained in the neighborhood table. The Discovery & Sync module performs relative slot synchronization (with a modified FTSP protocol) on the basis of asynchronous discovery (with Disco); these processes have rather low overhead.

The ChannelMonitor module realizes the bulk of the functionality of Chameleon, including the periodic measurment of $I(k)$ and the channel selection. It implements and provides the interface RadioControl for the purpose of transparently performing channel monitoring task, giving higher priority to O-MAC tasks with the radio resource. Chameleon only uses the radio when O-MAC is not occupying the resource. Whenever O-MAC attempts to start the radio, Chameleon immediately stops its monitoring task and returns the control of radio to O-MAC. ChannelMonitor also generates the channel schedule, which is input to the Listener module which implements the desired channel switching upon wakeup. In the diagram, colored components represent Chameleon modules which are modified or new with respect to the original O-MAC protocol. The Sender module is also modified to incorporate multichannel feature when transmission. The interfaces provided by Chameleon are MCM (Multi-Channel Monitoring) as follows.

```
interface MCM {
command  error_t  start ();
command  error_t  stop ();
command  uint8_t  getCh ();
command  void     setCh (k);
event    void     setChDone (error);
command  ch_arr   chVector ();
command  void     setUpdateInterval (t); }
```

Command getCh returns the index of the channel to use for communication based on recent channel monitoring result. The returned value of this command is included in the beacon sent out when the receiver wakes up. Command setCh is called to switch the channel for data transmission and the setChDone event is signaled after radio has stabilized on the new channel. chVector returns the current channel allocation in an array as V, while command setUpdateInterval provides a way for the application to adjust the update interval of channel assignment.

4 Protocol Evaluation

We evaluated Chameleon via both simulations, in Matlab, and experiments, based on an implementation in TinyOS 2.x for the TelosB platform[1]. We show, using simulation[12], that the performance of not only the metric w but also Chameleon compares favorably with other multi-channel MAC protocols under various traffic scenarios and network topologies. To validate Chameleon's performance in the presence of a realistic environment and (TelosB) platform effects, we experimentally evaluated three main metrics, namely, the end-to-end delivery ratio, the average receive duty cycle, and energy efficiency, of Chameleon with other benchmark protocols under various circumstances.

Delivery ratio is computed periodically, i.e., the number of successfully received packets at destinations divided by the number of packets attempted to be sent from sources. Due to the receiver-centric nature of these multi-channel protocols, we only consider the receive duty cycle at a node, which is represented by the fraction of active periods for listen or receive to the total period of time. The transmit duty cycle is approximately equal to the receive duty cycle because both Y-MAC and Chameleon are synchronous protocols. Given that data period of each slot takes t time, the energy efficiency is t multiplied by the number of slots that received packets successfully divided by the total active time for listening or receiving. We likewise corroborated its ability to tolerate external traffic and its relative improvements over both single channel (specifically, BoX-MAC and O-MAC) and multi-channel protocols (MMSN and Y-MAC).

Towards comparing with the other two multi-channel protocols in a fair manner, we made several necessary modifications to MMSN [18]. The frequency assignment of MMSN evenly allocates the available channels to neighbors. For media access, MMSN as specified does not consider duty cycling. As in Chameleon, however, we let each receiver listen at its own slot, and thus avoid the more expensive frequency toggle preamble incurred in the original specification of MMSN, given that senders are aware of receiver schedule. In other words, the modified version of MMSN that implemented has reduced protocol overhead. We implemented the modified MMSN and Y-MAC on TelosB platform. Based on current implementation of O-MAC, the average slot length of Chameleon is 16ms (same as O-MAC) and of Y-MAC is 20ms; the latter is larger since channel switching (and synchronization) is performed in every slot. MMSN operates at full duty cycle as in its original specification. The data packet size is fixed at 60 bytes. All data communications are performed in unicast mode. The size of the backoff window in each slot is 4ms. Neighbor discovery and time synchronization services are provided by O-MAC. In the comparison, we did not let Chameleon enforce restrictions on incoming traffic even if all channel capacities had been exceeded. (Such policing would, however, help the performance of Chameleon.) The monitoring overhead is zero for both MMSN and Y-MAC since channel assignment is done either a priori or deterministically; and around 0.6% duty cycle for Chameleon under three channels.

As for the single channel protocols, we used existing implementation of BoX-MAC, which is representative of duty-cycled asynchronous protocols, and O-MAC, which is representative of duty-cycled synchronous protocols. BoX-MAC [13] is the default low power listening protocol implemented in TinyOS-2.x. We let its receive check interval be set to 100ms.

Metrics for a Clique Network. Our first experiment was repeated for the five protocols in a clique network whose traffic load increases over time. The load increases adding independent flows to the network, with no flow sharing a source or a destination node with any other flow. Flows have one of three rates, with 1 packet every 100 milliseconds or 50 milliseconds, or 25 milliseconds, resulting in a load of approximately 10%, 20%, or 40% duty cycle, respectively. 6 independent flows are successively added in the network, with loads of 10%, 20%, 40%, 40%, 10%, and 20% respectively.

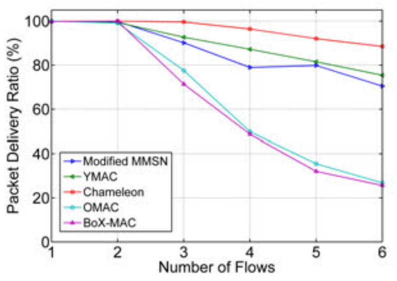
To avoid experimental error due to external interference from the environment, we collected measurements on the noise level for every available channel in our testbed. This gave us three relatively free channels in our testbed for this experiment, i.e., channels 22, 24, and 18. (Note that although there are 16 channels available on TelosB platform, it has been shown that adjacent channels actually interfere with each other [8]. Therefore, we avoided using adjacent channels in all our experiments.)

Fig. 4 (a)(b)(c) plot the metrics for these five protocols. We see that single channel protocols are much more negatively affected by the augmentation of traffic load than are multi-channel protocols. The packet delivery ratio of O-MAC is only slightly higher than that of BoX-MAC, but the duty cycle of BoX-MAC is 2 to 4 fold of O-MAC, suggesting that synchronous receiver-centric MAC protocol may be substantially more energy-efficient than asynchronous sender-centric protocols. The efficiency of both protocols decreases significantly as the traffic load increases. We also see that the overhead involved in Chameleon over O-MAC is within a 1% duty cycle.

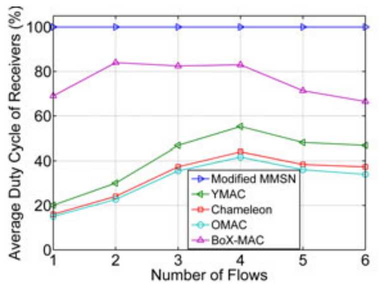
Chameleon maintains the highest delivery ratio of the three multichannel protocols as the traffic loads increases. In comparison with Y-MAC, MMSN has a worse delivery ratio because channel 22 is overloaded with flows (3 receivers are statically assigned to the same channel). Fig. 4(b) shows the average duty cycle of the receiver, which is proportional to the average traffic load. Y-MAC incurs about 10% higher overhead than Chameleon due to its continuous channel switching scheme. On the other hand, the primary overhead of Chameleon—channel monitoring—involves insignificant energy consumption. Fig. 4(c) illustrates the overall energy efficiency of each protocol. Chameleon has 62% to 55% efficiency as internal network load grows, which is on average 40% and 20% more efficient than modified MMSN and Y-MAC, respectively.

Metrics for a Clique Network with External Interference. Static assignment of load to channels, as in Y-MAC and MMSN, is inherently inefficient if the utilization of the shared spectrum by external systems is not monitored. Since Chameleon monitors channels comprehensively, it is intrinsically adaptable to dynamic and unknown wireless environments. Our next experiment introduced an external interferer to the network. In this experiment, 3 flows with duty cycles of 10%, 20%, and 40% exist in the network, and they use 3 of available channels. Time is divided into 8 periods. In period 1, there is no external interferer. During times 2 to 4, the interferer transmits on channel 18 with loads of 20%, 40%, and 60% sequentially. Later, interferers switch to channel 22 at time 5 and repeat the same increasing load pattern on channel 22.

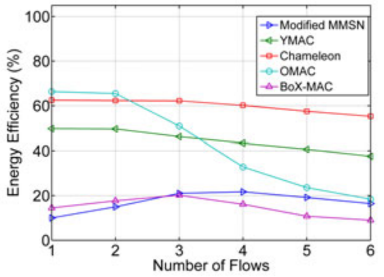
Fig. 4(d)(e)(f) shows the resulting delivery ratio, mean receive duty cycle, and energy efficiency. Initially, Chameleon and MMSN both distribute three flows into the three channels while Y-MAC evenly allocates traffic onto every channel. When the interferer on channel 18 increases its load, both MMSN and Y-MAC retain their current channel usage resulting in a reduced delivery ratio. In constrast, Chameleon detects the interference level increase on channel 18 and moves its traffic to other better channels. Thus, a high packet reception ratio as well as high energy efficiency is maintained by Chameleon's channel allocation scheme.



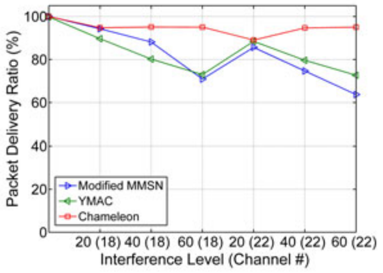
(a) Average Packet Delivery Ratio



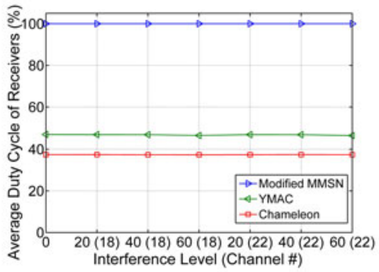
(b) Average Duty Cycle



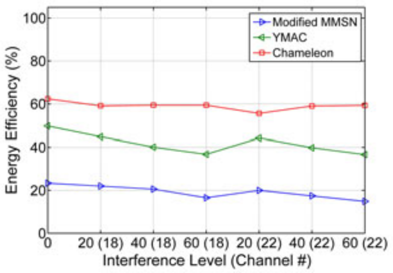
(c) Energy Efficiency



(d) Average Packet Delivery Ratio



(e) Average Duty Cycle



(f) Energy Efficiency

Fig. 4. (a)(b)(c) The Number of Internal Network Flows Increases in an Experimental Clique Network, and (d)(e)(f) External Interference Load Changes in an Experimental Clique Network

5 Related Work

The state-of-the-art in research includes a significant number of multi-channel MAC protocols for sensor networks. Per our earlier classification, the first category statically assigns multiple frequencies to nodes in the network as a way of topology control, in order to reduce potential interferences. Channel allocation is carried out beforehand, and is independent of real traffic conditions, such as in [18][16]. In [18], every node is assigned a channel for data reception such that most of two-hop neighbors do not communicate on the same channel. The TMCP protocol [16] divides nodes into several subsets of different channels, wherein nodes only communicate within their subset for simplicity of implementation. These schemes require a centralized channel assignment algorithm to execute in the beginning and the channel utilization is not adjusted according to communication load or interference on each channel.

Another approach expands the set of channels being used when the contention on the current channel become higher than an empirically chosen threshold. A distributed protocol in [11] lets all nodes in the network start in their home channel. When the channel becomes overloaded, a fraction of the nodes migrate to the next one. Channel switching is performed with a probability such that while alleviating congestion, it avoids having all nodes jump to the new channel. However, this protocol does not have a global view of the quality of each channel, thus, channel switching need not result in higher efficiency. Another work [10] presents a centralized protocol for load balancing across channels for throughput maximization. Each node periodically decides which channel to use based on measurements from the base station. The authors assume that network throughput is optimized as long as loads are distributed equally on each channel. There are also schemes based on frequency hopping [3] which are designed mainly for wireless ad hoc networks, which involve continuous switching of channel from slot to slot even when there is no need for transmission.

Few MAC protocols explicitly design multichannel scheduling with duty cycling to achieve high energy efficiency, which is the focus of this paper. A relatively recent multi-channel protocol, Y-MAC [9], exploits both duty cycling and multi-channel utilization. Every receiver wakes up at its non-overlap slot within each frame on home channel. If more packets need to be received, the receiver will stay awake but hop to the next channel for reception. The merit of this scheme lies in its staggered non-overlapping channel utilization over the extended M slots, while its weakness is that contiguous channel switching is expensive and the non-overlapping is guaranteed only within the M slots.

6 Concluding Remarks

This paper presented a new multi-channel MAC protocol, Chameleon, for duty-cycled wireless sensor networks. Chameleon betters the energy efficiency of existing protocols by adapting both the duty cycle and the channels that are being used. On one hand, it attempts to maximize spectrum utilization, via a light-weight channel utilization metric w that lets it split loads across channels effectively. On the other hand, it uses a receiver-centric approach to minimize on-duty time at the receiver, while letting senders wakeup only when they need to send and know the receiver is awake.

Experimental results confirm that Chameleon enhances energy efficiency substantially as compared to other multi-channel protocols under various internal traffic scenarios. Related experiments have shown us that external interference in long-lived WSNs is nontrivial, and is also typically unpredictable. Chameleon naturally coexists with dynamic conditions in spectrum and improves energy efficiency to a large extent.

The current design of Chameleon has not involved the broadcast scenario, which will be extended in the future. Future work will also examine the dynamics of Chameleon under different network topologies. We seek to address potential stabilization issues in channel selection via lightweight coordination among receivers.

References

1. CC2420 Datasheet, `http://www.ti.com`
2. Ahmed, N., Kanhere, S., Jha, S.: Multi-channel interference measurements for wireless sensor networks (poster). In: IPSN
3. Bahl, P., Chandra, R.: SSCH: Slotted seeded channel hopping for capacity improvement in ieee 802.11 ad-hoc wireless networks. In: MobiCom, pp. 216–230 (2004)
4. Buettner, M., Yee, G., Anderson, E., Han, R.: X-MAC: A short preamble mac protocol for duty-cycled wireless sensor network
5. Cao, H., Parker, K.W., Arora, A.: O-MAC: A receiver centric power management protocol. In: The 14th IEEE International Conference on Network Protocols, ICNP (2006)
6. Dutta, P., Culler, D.: Practical asynchronous neighbor discovery and rendezvous for mobile sensing applications. In: SenSys, pp. 71–84 (2008)
7. Halkes, G.P., Langendoen, K.G.: Crankshaft: An Energy-Efficient MAC-Protocol for Dense Wireless Sensor Networks. In: Langendoen, K.G., Voigt, T. (eds.) EWSN 2007. LNCS, vol. 4373, pp. 228–244. Springer, Heidelberg (2007)
8. Jain, N., Das, S.R., Nasipuri, A.: A multichannel csma mac protocol with receiver-based channel selection for multihop wireless networks. In: Proceedings of IEEE, IC3N (2001)
9. Kim, Y., Shin, H., Cha, H.: Y-MAC: An energy-efficient multi-channel mac protocol for dense wireless sensor networks. In: IPSN, pp. 53–63 (2008)
10. Le, H.K., Henriksson, D., Abdelzaher, T.: A control theory approach to throughput optimization in multi-channel collection sensor networks. In: The 6th International Conference on Information Processing in Sensor Networks, IPSN (2007)
11. Le, H.K., Henriksson, D., Abdelzaher, T.: A practical multi-channel media access control protocol for wireless sensor networks. In: IPSN, pp. 70–81 (2008)
12. Li, J., Arora, A.: Chameleon. Technical Report OSU-CISRC-11/09-TR52, The Ohio State University, CSE (2009)
13. Moss, D., Levis, P.: BoX-MACs: Exploiting physical and link layer boundaries in low-power networking. In: Technical Report SING-08-00
14. So, J., Vaidya, N.: Multi-channel mac for ad hoc networks: Handling multi-channel hidden terminals using a single transceiver. In: ACM MobiHoc (2004)
15. Sun, Y., Gurewitz, O., Johnson, D.B.: RI-MAC: A receiver-initiated asynchronous duty cycle mac protocol for dynamic traffic loads in wireless sensor networks. In: SenSys, pp. 1–14 (2008)
16. Wu, Y., Stankovic, J.A., He, T., Lin, S.: Realistic and efficient multi-channel communications in wireless sensor networks. In: Infocom, pp. 1193–1201 (2008)
17. Zhang, J., Zhou, G., Huang, C., Son, S.H., Stankovic, J.A.: TMMAC: An energy efficient multi-channel mac protocol for ad hoc networks. In: IEEE ICC (2007)
18. Zhou, G., Huang, C., Yan, T., He, T., Stankovic, J., Abdelzaher, T.: MMSN: Multi-frequency media access control for wireless sensor networks. In: The 25th IEEE International Conference on Computer Communication, Infocom (2006)