CMAC: An Energy Efficient MAC Layer Protocol Using Convergent Packet Forwarding for Wireless Sensor Networks

Sha Liu, Kai-Wei Fan and Prasun Sinha
Department of Computer Science and Engineering, The Ohio State University
Email: {liusha,fank,prasun}@cse.ohio-state.edu

Abstract— **Low duty cycle operation is critical to conserve energy in wireless sensor networks. Traditional wake-up scheduling approaches either require periodic synchronization messages or incur high packet delivery latency due to the lack of any synchronization. In this paper, we present the design of a new low duty-cycle MAC layer protocol called Convergent MAC (CMAC). CMAC avoids synchronization overhead while supporting low latency. By using zero communication when there is no traffic, CMAC allows operation at very low duty cycles. When carrying traffic, CMAC first uses anycast to wake up forwarding nodes, and then converges from route-suboptimal anycast with unsynchronized duty cycling to route-optimal unicast with synchronized scheduling. To validate our design and provide a usable module for the community, we implement CMAC in TinyOS and evaluate it on the Kansei testbed consisting of 105 XSM nodes. The results show that CMAC at 1% duty cycle significantly outperforms BMAC at 1% in terms of latency, throughput and energy efficiency. We also compare CMAC with other protocols using simulations. The results show for 1% duty cycle, CMAC exhibits similar throughput and latency as CSMA/CA using much less energy, and outperforms SMAC and GeRaF in all aspects.**

I. INTRODUCTION

Extending the lifetime of battery powered wireless sensor networks is critical because touching the sensor nodes for replacement might be expensive or even impossible. Since idle listening consumes almost the same energy as receiving or transmitting, duty cycling the radio is important to achieve long lifetime. However, a low duty cycle usually causes performance degradation in throughput and latency which are critical metrics for various applications such as event tracking and surveillance. These conflicting objectives motivate the design of a new MAC layer protocol called **Convergent MAC (CMAC)**. Compared to other MAC layer protocols like BMAC [1] and SMAC [2], CMAC can significantly reduce latency and improve throughput while supporting very low duty cycles.

Current duty cycling MAC layer protocols for wireless sensor networks are either synchronized using explicit schedule exchanges or totally unsynchronized. However, both have their weaknesses and deficiencies. SMAC [2], TMAC [3] and DMAC [4] use periodic synchronization messages to schedule duty cycling and packet transmissions. Such message exchanges consume significant energy even when no traffic is present. BMAC [1] uses unsynchronized duty cycling and uses long preambles to wake up receivers. However, the long preamble mechanism has following problems. First, the latency accumulated along multihop routes could be overwhelming due to the use of long preambles on each hop. Second, the energy consumed on preamble transmission and reception after the receiver has woken up is wasted. This is due to lack of information at the sender side about the wake-up schedule of the receiver, and thus the preamble length is chosen conservatively. Third, neighbor nodes other than the intended receiver will also be kept awake by the long preamble until the data packet transmission finishes, which is also wasteful since they are doing unneeded preamble overhearing. Polastre et. al. propose a link abstraction called Sensornet Protocol (SP) [5] to adjust the preamble length by observing recent and nearby traffic. However, SP still relies on long preambles to initiate data flows, and it cannot dynamically select the next hop if the intended next hop is currently not available because of sleeping or interference.

The above problems motivate our design of an energy efficient MAC layer protocol called Convergent MAC (CMAC). CMAC uses unsynchronized sleep scheduling like BMAC when there is no packet to transmit. While transmitting packets, CMAC first uses *aggressive RTS* (Section II-A) to *anycast* (Section II-B) packets to potential forwarders which wake up first and detect

the traffic using *double channel check* (Section II-A). Once the sender is able to transmit packets to a node with acceptable routing metric, CMAC *converges from anycast forwarding to unicast* (Section II-C) to avoid the overhead of anycast. To validate the practicability of CMAC, we implement CMAC in TinyOS [6] and compare it with BMAC on the Kansei testbed [7]. We also evaluate CMAC in *ns2* [8] against SMAC, a GeRaF variant, and 802.11 based CSMA/CA protocol. The results show CMAC outperformns other duty cycle scheduling protocols in all aspects while providing comparable throughput and latency performance as fully awake CSMA protocol.

The main contributions of this paper are listed below:

- We propose CMAC, a novel MAC layer protocol, which improves the latency and energy efficiency by utilizing proposed aggressive RTS, anycast and convergent packet forwarding mechanisms.
- We analytically model the performance comparison of anycast and unicast, and the performance of convergent packet forwarding;
- We present details of the implementation and real field evaluation of CMAC to validate our design goals.

The rest of the paper is organized as follows. Section II presents the design of CMAC. Section III presents our implementation and real field experimental results on Kansei testbed comparing CMAC with BMAC. Section IV presents simulation results comparing CMAC with SMAC, CSMA/CA and a GeRaF variant. Section V summarizes the literature and compares CMAC with them. Finally, Section VI concludes the paper.

II. CONVERGENT MAC (CMAC)

Motivated by the limitations of current approaches, we propose a new MAC layer protocol called **Convergent MAC (CMAC)** that supports low latency and high throughput as well as low duty cycle operation. CMAC has three main components: *Aggressive RTS* equipped with *double channel check* for channel assessment, *anycast* to quickly discover forwarder, and *convergent packet forwarding* to reduce the anycast overhead.

A. Aggressive RTS

The long preamble mechanism of BMAC incurs high latency in order to ensure that the receiver is awake before sending the data. However, the receiver may wake up much earlier than the end of the preamble, which makes part of the preamble transmission wasteful. We
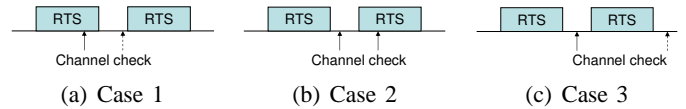

Fig. 1. Double Channel Check used by CMAC. (a) The first check detects the RTS burst. (b) The second check detects the RTS burst. (c) impossible if RTS length is chosen carefully.

propose to use *aggressive RTS* to replace the long preamble, which breaks up the long preamble into multiple RTS packets (also called an *RTS burst*). The RTS packets do not use long preambles, and are separated by fixed short gaps each of which allows receivers to send back CTS packets. Once the transmitter receives a CTS, it sends the data packet immediately. Each gap need not accommodate an entire CTS transmission as long as the RTS sender can detect the preamble and cancel the next RTS transmission accordingly. The number of RTS packets to be sent in one RTS burst depends on the duty cycle length. For the same duty cycle length, the duration of one RTS burst is roughly the same as the long preamble used by BMAC. If nodes uniformly randomly wake up during each duty cycle, the expected latency at each hop could be reduced by half.

To allow nodes to work at a very low duty cycle, nodes must assess the channel very quickly each time they wake up. However, if the receiver wakes up during the gap between two RTS transmissions, it may miss this RTS burst. So we propose to use *double channel check* which works by assessing the channel twice with a fix short separation between them each time a node wakes up. For each channel check, nodes sample the channel for up to 5 times. Between these two channel checks, the radio could be put to sleep mode to save energy. If the first check detects a busy channel, the second check will be canceled (Fig. 1(a)). Otherwise, the second check is performed (Fig. 1(b)). The positive conclusion on busy channel from either check will keep the node awake anticipating an RTS. To prevent the scenario in Fig. 1(c), the interval must be shorter than the RTS transmission time. This can be satisfied by padding RTS packets with extra bytes if needed. (We discuss the choice of these parameters In our implementation in Section III.) Such a "double-check" mechanism ensures that nodes will not miss any nearby RTS burst.

The cooperation of aggressive RTS and double channel check require both the gap between two RTS and the interval between two channel checks to be fixed. CMAC achieves this by sending all RTS packets without assessing the channel except the first one. This may cause two RTS bursts from more than one transmitter to
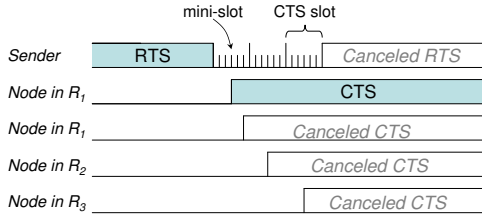
Fig. 2. CTS contention resolution. The first CTS cancels others.

Fig. 3. Example of cost region generation in CMAC using geographical distance as routing metric.

collide, in which case nodes need to retransmit the RTS burst from the beginning after current collisions have been resolved.

B. Anycast Based Forwarding

Aggressive RTS can reduce per-hop latency by half on average for unicast. However, if nodes other than the unicast target can also reply to the transmitter, per-hop latency could be further reduced since some routing progress could be made while the target nexthop is still asleep. We define the neighbor nodes of the transmitter that are closer to the destination as a *forwarding set*. Simple calculation shows for duty cycle length 1 and forwarding set size n, it takes on average $\frac{1}{n+1}$ to get in contact with at least one of them.

However, more than one node in the forwarding set may try to reply to the same RTS, and the one closest to the destination should be elected to receive the data packet. In CMAC, the CTS transmissions are prioritized according to the routing metrics of contending nodes. Nodes with better routing metrics can send CTS packets earlier, while other overhearing nodes cancel their CTS transmissions accordingly, and nodes that can make little progress could be excluded. The routing metric used could be very general, such as geographical distance, hop count, ETX [9], ETT [10] and PRR×Dist [11]. In this paper, we only investigate the use of geographical distance to resolve CTS contentions.

CMAC partitions the forwarding region into 3 subregions, R_1, R_2, and R_3, such that nodes in R_i are closer to the destination than nodes in R_j for $i < j$ (Fig. 3). Each gap between two consecutive RTS packets is divided into 3 sub-intervals called *CTS slots*. Nodes in region closer to the destination can send CTS packets in earlier CTS slots. Each CTS slot is further divided into several *mini-slots* to resolve the contention within each region, and each receiver will randomly choose one mini-slot to start its CTS transmission (Fig. 2). On detecting busy channel, pending CTS transmissions will be canceled assuming the existence of another CTS.

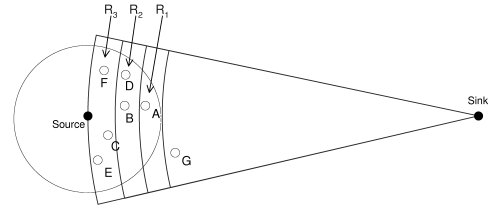*1) Performance Analysis of Anycast :* Anycast can lock a forwarding candidate node faster than unicast, but this is achieved at the cost of higher overhead and less routing progress for each individual transmission. In this subsection, we analytically model the performance of anycast. The metric used in this analysis is the latency of each transmission normalized by its geographical routing progress called *normalized latency*. For the rest of the analysis, the length of a duty cycle is normalized to 1, and the notations used are summarized as follows.

- L: normalized latency.
- ρ: node density.
- S: area of the forwarding region.
- X: geographical progress made by anycast.
- Y: the latency of finding the first awake node in the forwarding set. Then its CDF is $F(y) = 1-(1-x)^n$, and it is independent of X.
- r: transmission range.
- r_0: the minimum progress required for a neighbor node to be present in the forwarding set.
- d: distance from the transmitter to the destination.

Note that lower duty cycle leads to longer duty cycle length since the time to check the channel is fixed. Then for very low duty cycles, the RTS and data packet transmission times could be ignored. Hence, $E[Y] \approx \int_0^1 y dF(y) = \frac{1}{\rho S+1}$. Then the expected normalized latency could be expressed as

$$E[L] = E[\frac{Y}{X}] = E[Y]E[\frac{1}{X}] = \frac{1}{\rho S + 1}E[\frac{1}{X}], \quad (1)$$

where the second equality is due to the independence.

To compute $E[\frac{1}{X}]$, we can simply consider the upper half of the forwarding region as region OAR in Fig. 4. For any point B between O and R with x coordinate no less than r_0, on segment OR, its weight is $\frac{1}{x}$ times the length of arc $\widehat{BC}$ (Fig. 4), while the length of $\widehat{BC}$ is

$$
\begin{aligned}
|\widehat{BC}| &= |CD| \times \angle CDO & (2) \\
&= |CD| \arccos \frac{|CD|^2 + |OD|^2 - |OC|^2}{2|CD||OD|} & (3) \\
&= (d-x) \arccos \frac{(d-x)^2 + d^2 - r^2}{2d(d-x)}. & (4)
\end{aligned}
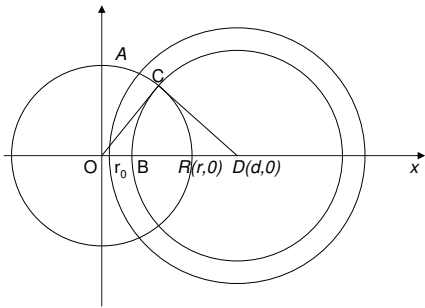$$

3

Fig. 4. Calculation of $E[\frac{1}{X}]$. O is the sender, D is the destination.

Hence,

$$
\begin{aligned}
S &= \int_{r_0}^{r} |\widehat{BC}| dx \qquad (5) \\
&= \int_{r_0}^{r} (d-x) \arccos \frac{(d-x)^2 + d^2 - r^2}{2d(d-x)} dx, \quad (6)
\end{aligned}
$$

and

$$
\begin{aligned}
E[\frac{1}{X}] &= \frac{\int_{r_0}^{r} \frac{1}{x}|\widehat{BC}| dx}{S} \qquad (7) \\
&= \frac{\int_{r_0}^{r} \frac{1}{x}(d-x)\arccos \frac{(d-x)^2+d^2-r^2}{2d(d-x)} dx}{\int_{r_0}^{r} (d-x)\arccos \frac{(d-x)^2+d^2-r^2}{2d(d-x)} dx}. \quad (8)
\end{aligned}
$$

There are three parameters affecting $E[L]$: r_0, S and ρ. S depends on r_0 and d. Fig. 5(a) plots $E[L]$ versus r_0 for different d and ρ values. It can be seen that for a certain node density, d affects $E[L]$ only a little, but higher node density clearly leads to smaller $E[L]$. In addition, for certain d and node density, there is an optimal value of r_0 to optimize $E[L]$.

After finding the optimal r_0, nodes still need to decide to use anycast or unicast. For unicast, the normalized latency is bounded by $\frac{1}{2r}$. Hence for anycast to be superior than unicast on average, it should have lower expected normalized latency. Using Equation (1), this criterion leads to the following critical node density above which anycast is better

$$
\rho > \frac{2rE[\frac{1}{X}] - 1}{S}. \qquad (9)
$$

According to the above expression, the minimum node density depends on the $\frac{d}{r}$ ratio and r_0. For r_0 between 1% and 50%, we plot the critical node densities versus different $\frac{d}{r}$ ratios in Fig. 5(b). Hence if the minimum density is lower than the actual density, anycast should be used, and unicast should be used otherwise.

The above formulation also leads to a localized technique to determine if anycast will lead to better normalized latency. Specifically, for a node with n neighbor nodes in its forwarding set with each node making r_i progress ($1 \le i \le n$), it can decide if anycast is better if

$$
\frac{1}{n(n+1)} \sum_{i=1}^{n} \frac{1}{r_i} < \frac{1}{2 \max_{1 \le i \le n}\{r\}} \qquad (10)
$$

C. Converging from Anycast to Unicast

Although anycast obviates the need for synchronization messages and has better chance to make progress in packet forwardings than unicast, it has two main shortcomings. First, anycast may choose suboptimal routes because the best next hop is sleeping or due to interference. Second, the overhead of anycast RTS/CTS exchange is usually higher than its unicast counterpart. Hence, a mechanism is needed to reduce the overhead incurred by anycast, and we propose *convergent packet forwarding* to resolve these problems as follows.

In CMAC, the node will remain awake for a short duration after receiving a data packet. During this period, a node with better routing metric could wake up and become the receiver of the next anycast. If the latest anycast receiver has a routing metric close to the best, CMAC will use unicast instead to avoid the overhead of anycasting. Taking Fig. 3 as an example, node C might be the first one to wake up and participate in the anycast, and then node B followed by node A. After A receives the anycast, the transmitter starts to unicast to A, finishing the convergence. However, it is possible that nodes with acceptable routing metrics may not exist. For example, this will happen if there is no node in region R_1 in Fig. 3. Hence if the transmitter cannot find a better next hop than the current one after a duty cycle length, it switches to unicasting. In this way, the packet forwarding converges from anycast to unicast for each link. After some time without successful data packet reception, CMAC will timeout and nodes will again start following unsynchronized idle duty cycles.

To reduce the packet delivery latency, a long awake duration after receiving each packet is preferred. But longer awake period also leads to more energy consumption. Hence, nodes need to optimize the length of this period based on the observed traffic information such as average packet arrival rate to accommodate latency or energy consumption requirements. In what follows, we use a simple model to analyze the latency and power consumption under different awake durations. The duty cycle length is again normalized to 1, and other denotations are listed below.

- A: active duration after receiving a packet.
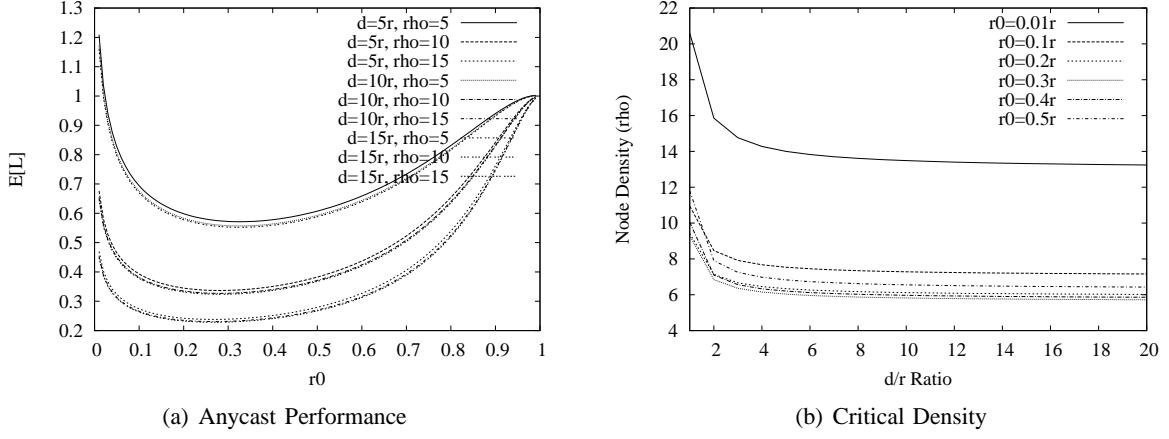- Z: packet arrival interval (a random variable).

4

(a) Anycast Performance   (b) Critical Density

Fig. 5. Numerical results for anycast performance analysis.

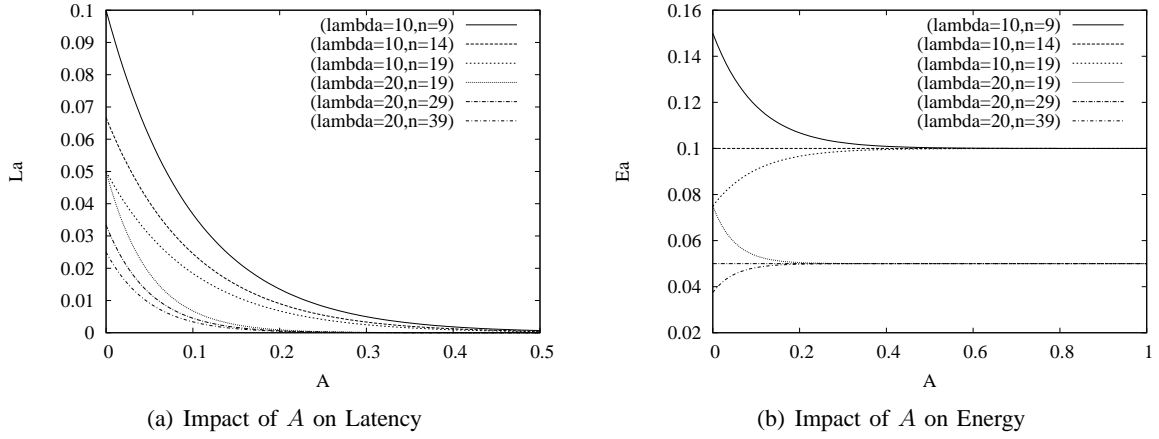(a) Impact of A on Latency   (b) Impact of A on Energy

Fig. 6. Numerical results for convergence performance analysis.

- G: the CDF of Z.
- λ: average packet arrival rate.
- p_r: power for idle listening and receiving.
- p_t: power for transmitting.

If the next packet arrives at the sender before the active duration A timeouts, which happens with probability $P\{Z < A\} = G(A)$, unicast will be used. Hence the latency is 0 (before transmitting the data packet), and the idle listening of length Z contributes to the entire energy consumption. Otherwise, if the next packet arrives at the sender before the active duration timeouts, which happens with probability $P\{Z \geq A\} = 1 - G(A)$, anycast will be used. Hence the average latency is the average time needed to contact at least one receiver which is $\frac{1}{n+1}$, and the average energy consumption has two components, idle listening for duration A and transmitting aggressive RTS for $\frac{1}{n+1}$. Therefore, the average latency L_a and average energy consumption E_a

are

$$
\begin{aligned}
L_a &= \frac{1}{n+1}(1 - G(A)), \\
E_a &= G(A)p_r E[z|z < A] \\
&\quad + (1 - G(A))(p_r A + p_t \frac{1}{n+1}).
\end{aligned}
$$

If the packet arrival process is Poisson with parameter λ, then $G(A) = 1 - e^{-\lambda A}$, and

$$
E[z|z < A] = \frac{\int_0^A z\lambda e^{-\lambda z}dz}{\int_0^A \lambda e^{-\lambda z}dz} = \frac{\frac{1}{\lambda} - (A + \frac{1}{\lambda})e^{-\lambda A}}{1 - e^{-\lambda A}}.
\tag{11}
$$

Hence,

$$
L_a = \frac{e^{-\lambda A}}{n+1}, \tag{12}
$$

$$
E_a = \frac{p_r}{\lambda} + (\frac{p_t}{n+1} - \frac{p_r}{\lambda})e^{-\lambda A}. \tag{13}
$$

We plot in Fig. 6(a) and Fig. 6(b), L_a and E_a versus A for different λ and n ($p_r = 1$ and $p_t = 1.5$). It can be

seen that L_a decreases with the increase of A given a certain n, but E_a has more complex variation patterns. Using Equation (13) we can see that there are three cases as follows.

1) If $\lambda < \frac{p_r(n+1)}{p_t}$, E_a increases with A up to $\frac{p_r}{\lambda}$.
2) If $\lambda = \frac{p_r(n+1)}{p_t}$, $E_a = \frac{p_r}{\lambda}$.
3) If $\lambda > \frac{p_r(n+1)}{p_t}$, E_a decreases with A down to $\frac{p_r}{\lambda}$.

Hence, for high packet arrival rate with certain node densities, nodes can increase the awake period without worrying about the latency and energy performance. But for low traffic scenarios, the determination of A is application dependent.

The unicast after such convergence may or may not use normal RTS/CTS. In our experiments, CMAC does not use RTS/CTS after convergence for fair comparison with BMAC. In our simulations, CMAC uses normal RTS/CTS that is similar as 802.11 after convergence for comparison with 802.11, SMAC and GeRaF.

If the event moves fast, the source nodes may continuously change with each of them generating only a small number of packets. In this case, the convergence may still happen at places closer to the destination where the routes may be more stable. For some other cases such as low data rates, the convergence may not happen, but CMAC can still use aggressive RTS and anycast to make quick progress towards the sink.

D. Synchronized Wake-up Schedule

In order to save more energy after convergence, nodes can synchronize with their neighbor nodes to use some kind of wake-up schedule instead of keeping fully awake. In our simulations, we evaluate a CMAC variant using a staggered scheduling similar to DMAC [4] after convergence. When the transmitter intends to converge from anycast to unicast, it synchronizes its schedule with the receiver. The two nodes will maintain the staggered schedule as long as there is traffic between them. After a certain duration without traffic, the nodes go back to using unsynchronized duty cycling. This CMAC variant is described in more details in our technical report [12].

III. EXPERIMENTAL EVALUATION

Our TinyOS [6] implementation[1] of CMAC is based on XSM [13] which is similar to Mica2 mote [14] in CC1000 radio [15] and processing board. We set the mini-slot length to the transmission time of 1 byte on CC1000 radio which is $416\mu s$, a period long enough to

[1]Code available at `http://www.cse.ohio-state.edu/~liusha/cmac`.

TABLE I
IMPLEMENTATION PARAMETERS

CTS-slot length	$7.488ms$
Number of CTS-slots	3
Mini-slot length	$416\mu s$
Number of mini-slots	6
RTS packet size	44 bytes
Double channel check interval	$10ms$

accommodate the propagation delay and busy channel detection (One channel sampling takes about 265μ to finish). Other parameters are summarized in Table I. The Kansei testbed consists of 105 XSM nodes forming a 15×7 topology with node separation of 3 feet. The transmission range is set to 4 rows/columns in the testbed. Each XSM node is attached to a Linux-based stargate [16] through which command messages are sent to trigger the generation of packets.

We evaluate the throughput, latency and energy efficiency of CMAC against BMAC for two basic event scenarios, static event and moving event. Throughput refers to the total number of packets received at the sink in 600 seconds, latency is the average delay experienced by a packet, and energy efficiency refers to the energy consumption of the entire network for delivering one 36-byte packet to the sink (called normalized energy).

Note that the double channel check almost doubles the times of channel sampling in BMAC. Thus CMAC consumes more energy on channel assessment than BMAC if the duty cycle length is the same. To be fair, we evaluate CMAC with duty cycle length double that of BMAC in this section. For example, if BMAC uses $300ms$ duty cycle length, CMAC will use $600ms$. Since using $300ms$ duty cycle length in BMAC is roughly 1% duty cycle, we denote it by BMAC 1%, and denote CMAC using $600ms$ duty cycle length as CMAC 1%. To provide the baseline for throughput and latency evaluation, we also gathered the data for BMAC and CMAC without duty cycling, denoted by BMAC 100% and CMAC 100% respectively.

A. Static Event Scenarios

In this set of experiments, we emulate an event happening at one corner of the testbed. The source node sends all packets to the sink located at the diagonally opposite corner. We vary the data rate at source nodes, and the results are shown in Fig. 7.

For low data rates (0.2 ~ 0.5 packets/sec.), both CMAC 1% and BMAC 1% can deliver all packets (Fig. 7(a)), but Fig. 7(b) shows that CMAC 1% exhibits better latency performance than BMAC 1% due to the
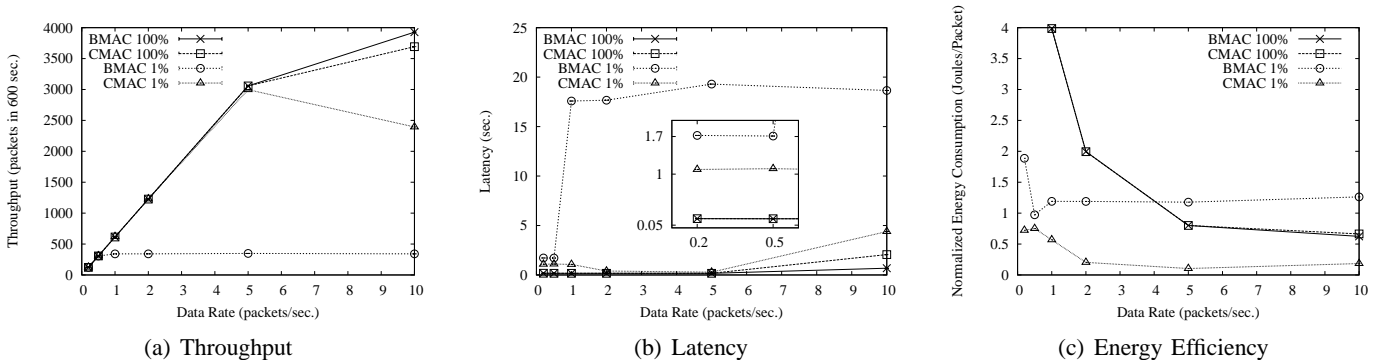
Fig. 7.   Experiment results of throughput, latency and energy efficiency performance of CMAC and BMAC under different data rates.

capability of aggressive RTS and anycast to discover awake potential forwarders.

Under high data rates (≥ 1 packet per second), BMAC 1% can not deliver all packets to the sink, and the flat curve shows that the channel capacity is reached due to the use of long preambles and multihop contention. CMAC 1% saves unnecessary long preambles, and thus not only significantly outperforms BMAC 1% but also provides similar throughput as BMAC 100% and CMAC 100% (Fig. 7(a)). In some cases, e.g., data rate of 2 and 5 packets per second, CMAC 1% even provides latency performance very close to that of BMAC 100% (Fig. 7(b)). This is due to the convergence of CMAC from anycast to unicast and the saving on anycast overhead. At the data rate of 10 packets per second, CMAC 1% does not provide throughput and latency very close to BMAC 100% or CMAC 100% because the high contention leads to some convergence duration times out which result in more RTS/CTS, but CMAC 1% still exhibits significant improvement over BMAC 1%.

Fig. 7(c) shows CMAC 1% utilizes the energy more efficiently than BMAC 1% and BMAC 100%, and the energy efficiency becomes better as the data rate increases. Hence, we conclude that CMAC is more suitable for providing high throughput and low latency while the idle duty cycle is low.

B. Moving Event Scenario

To evaluate the performance of CMAC for moving events, we let the the emulated event move along the bottom edge of the testbed at different speeds where faster speeds trigger more packets. The results of throughput, latency and energy efficiency are shown in Fig. 8.

Fig. 8(a) exhibits the advantage of CMAC 1% over BMAC 1% in throughput. The throughput of BMAC 1% increases with the increase of the moving speed for slow speeds, but it gradually drops after the moving speed

exceeds 1 row/sec. However, the throughput increase of CMAC 1% shows that it can accommodate the increased packet generation speed. Fig. 8(b) shows remarkable advantage of CMAC 1% over BMAC 1% in latency (less than $1s$ compared to more than $100s$). For BMAC 1%, the queueing delay contributes to most of the latency and is due to the use of long preambles. Fig. 8(c) shows the advantage of CMAC 1% in energy efficiency. CMAC 1% saves $75\% \sim 95\%$ normalized energy of BMAC 1%. In addition, the normalized energy consumption of CMAC 1% decreases gradually with the increase of moving speed because there are more chance for CMAC to converge when there are more active flows. But for BMAC 1%, the energy efficiency increases sharply due to the inefficiency of long preambles.

C. Anycast Performance

For low data rates, CMAC may not be able to converge from anycast to unicast because there isn't enough traffic. In such cases, the performance of CMAC depends on the aggressive RTS and anycast mechanism. Thus we evaluate the performance of the aggressive RTS and anycast mechanism in this section. The duty cycles are 1% and 0.1%, where each cycle is $3000ms$ and $6000ms$ respectively for BMAC 0.1% and CMAC 0.1%. The source node is located at one corner, and the sink is at the diagonally opposite corner. We vary the node density by adjusting the transmission range from 3 rows/columns to 8 rows/columns and run each experiment for 600 seconds. The data rate is chosen such that every packet is purely anycast enroute without any convergence or queuing delay. Due to the limited size of Kansei testbed, we present the latency normalized by the hop count of unicast, i.e., $\frac{Latency}{Hops}$, and the results are shown in Fig. 9(a) and 9(b) (Figures for throughput are omitted since all protocols can deliver all packets to the sink).

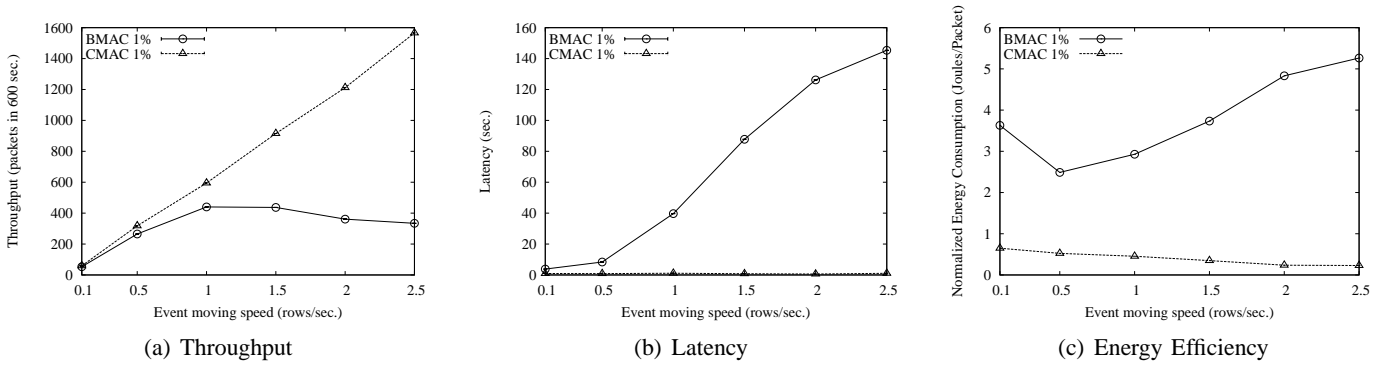CMAC reduces the latency of BMAC by about 33%

Fig. 8. Experiment results of throughput, latency and energy efficiency performance of CMAC and BMAC under different event moving speeds.

at both 1% and 0.1% duty cycles except for transmission range of 8 rows with 1% duty cycle, where the improvement is not very significant. The reason for this is that the packet can take as few as 2 hops to reach the destination while the last-hop transmission does not use anycast since the destination is already in range.

We also collect the route stretch of anycast, which is represented by the average number of hops of anycast normalized by the hop count of unicast. Fig. 9(c) shows CMAC 0.1% has larger stretch than 1%. This is because for higher duty cycles, the elected next hop is also better. As Fig. 9(a) and 9(b) show, even with route stretch, CMAC 1% can still outperform BMAC 1% due to the use of aggressive RTS and anycast.

IV. SIMULATION BASED EVALUATION

We also conduct simulations[2] for large networks to compare the throughput, latency and normalized energy consumption of CMAC with other protocols using *ns2* [8]. Our study is based on the following six protocols:

- **CSMA/CA:** Fully awake CSMA/CA.
- **Anycast:** Using the anycast mechanism described in Section II-B with radio fully awake.
- **GeRaF:** Using the anycast protocol in Section II-B with 10% duty cycle and $3ms$ active period, which is similar in essence to [17] [18].
- **CMAC:** Our proposed scheme described in Section II working on 1% idle duty cycle.
- **CMAC-S [12]:** Similar to CMAC, but a DMAC-like [4] staggered scheduling is used after convergence.
- **SMAC:** SMAC in [2] working at 10% duty cycle.

The simulations are conducted on a $2000m \times 2000m$ network with an event moving randomly at $10m/s$. We

[2]Code available at http://www.cse.ohio-state.edu/ ~liusha/cmac.

Tx range	$250m$	RTS size	14 bytes
Bandwidth	$38.4Kbps$	CTS size	14 bytes
Tx power	$27mA$	ACK size	28 bytes
Rx power	$10mA$	Data header	20 bytes
Idle power	$10mA$	Data payload	50 bytes
CTS slot	$0.2ms$	Anycast CTS	22 bytes
Active period	$3ms$	Preamble+PLCP	24 bytes

use $250m$ as the transmission range, but our protocol works for any radio transmission range. Other parameters are shown in Table II. In this section, we present results in mobile event scenarios for varying initial idle duty cycle, node density, and data rate. More simulation results are available in our technical report [12].

A. Initial Duty Cycle

First we evaluate the impact of the idle duty cycle by varying it from 0.1% to 1% for data rate of 10 packets/s. (GeRaF and SMAC use 1% to 10% duty cycles since they can barely deliver any packet for lower duty cycles.) Fig. 10 shows anycast and CSMA/CA have the best throughput performance, but CMAC can also provide comparable throughput while greatly exceeding SMAC and GeRaF even if the idle duty cycle is lower. In addition, CMAC provides latency performance comparable to CSMA/CA while using the least energy. Note that the energy efficiency in Fig. 10 only reflects the performance when there is traffic, and CMAC at low duty cycle actually consumes much less energy than CSMA/CA and anycast when the network is idle

B. Node Density

Next we evaluate the performance of CMAC in networks with different node densities. We vary the number of nodes in the network from 100 to 625 while keeping
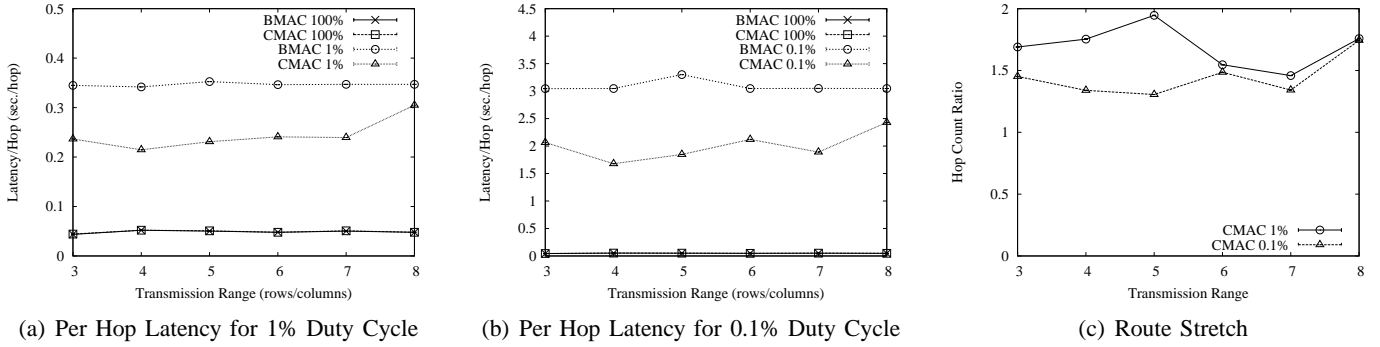
(a) Per Hop Latency for 1% Duty Cycle (b) Per Hop Latency for 0.1% Duty Cycle (c) Route Stretch

Fig. 9. Experiment results of anycast latency performance of CMAC 1% and CMAC 0.1% under different node densities.

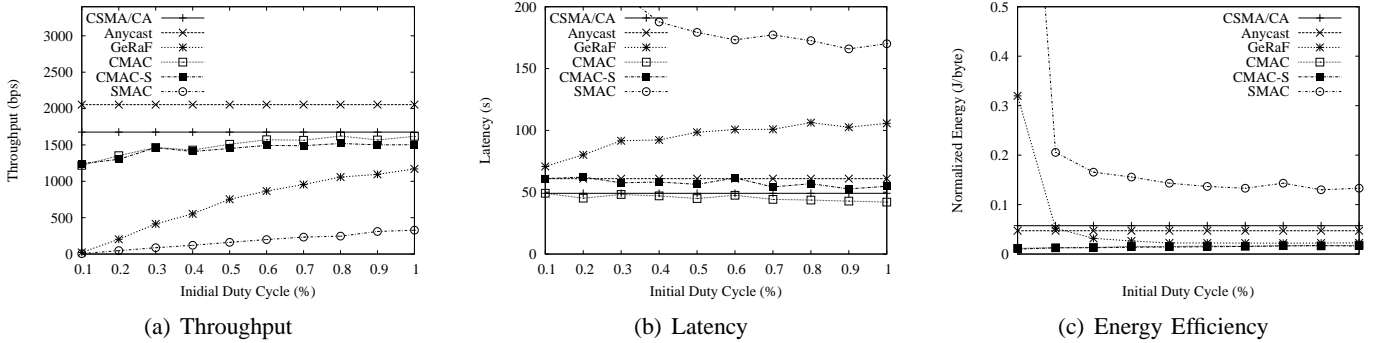(a) Throughput (b) Latency (c) Energy Efficiency

Fig. 10. Simulation results for throughput, latency and energy efficiency of CMAC, SMAC, GeRaF and CSMA/CA under different idle duty cycles. The duty cycles for GeRaF and SMAC are 1% to 10% due to their inability of delivering any packet for lower duty cycles.

the area and event size unchanged. Fig. 11 shows that the throughput, latency and normalized energy consumption all increase with the increase of node density. This is because more nodes are generating packets with higher node density. The throughput of anycast is the best because it can always take alternate path during high contention, while CMAC provides similar throughput as CSMA/CA (Fig. 11(a)). For latency, CMAC is also among the best. More importantly, CMAC outperforms all other protocols in normalized energy consumption.

C. Data Rate

Fig. 12 shows the simulation results of throughput, latency and normalized energy consumption of different protocols for different date reporting rates. CMAC and CMAC-S use the least energy, while achieving at least $90\% - 95\%$ of throughput of CSMA/CA.

V. RELATED WORK

In the context of energy efficient MAC layer designs, the proposed approaches can be broadly divided into two categories: synchronized and unsynchronized.

Synchronized MAC: Protocols using this mechanism require nodes to periodically synchronize wake-up schedules with their neighbors using explicit messages, and nodes wake up and sleep according to the synchronized schedules. SMAC [2], TMAC [3] and DMAC [4] fall into this category. This type of approaches requires periodic synchronization message exchange which consumes significant energy unnecessarily.

Unsynchronized MAC: Protocols in this category do not synchronize nodes until there is traffic. Usually this involves the use of long preambles like in BMAC [1] and SP [5]. However, the long preamble leads to high latency as explained in Section I. XMAC [19] tries to mitigate the negative impact of long preambles by using strobed preambles, which shares a similar idea with the Aggressive RTS used in CMAC. However, XMAC has a significantly longer awake period than CMAC due to the lack of a mechanisms like double channel check ($16ms$ on TelosB versus $6ms$ on XSM/Mica2 in CMAC).

MAC layer anycast is another way to avoid explicit synchronization [17] [18] [20] [21]. But when compared to CMAC and the Contention-Based Forwarding (CBF) mechanisms [22]–[24], their RTS/CTS exchange schemes are more complicated and inefficient. Both MAC layer anycast and CBF work by prioritiz-
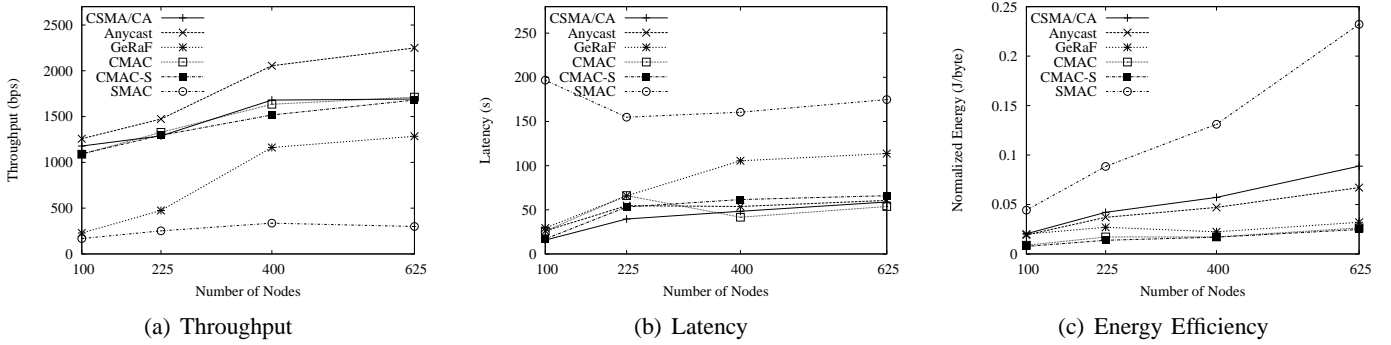
(a) Throughput (b) Latency (c) Energy Efficiency

Fig. 11. Simulation results for throughput, latency and energy efficiency performance of CMAC, SMAC, GeRaF and CSMA/CA under different node densities.
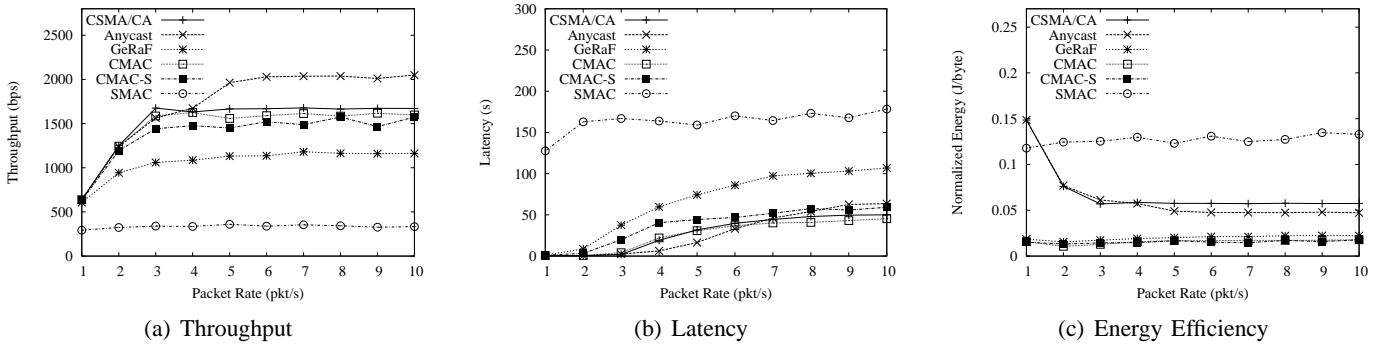

(a) Throughput (b) Latency (c) Energy Efficiency

Fig. 12. Simulation results for throughput, latency and energy efficiency performance of CMAC, SMAC, GeRaF and CSMA/CA under different data rates. CMAC and CMAC-S can achieve at least 90% to 95% throughput of CSMA/CA.

ing the CTS replying from potential forwarders, but CBF schemes use CSMA based contention among CTS repliers to resolve CTS collisions and thus have lower overhead. However, CBF mechanisms are only explored in the context of MANET where performance impact of the node mobility is the major concern. The design of CMAC, however, takes all aspects of designing a real protocol such as sleep scheduling and channel check into consideration. Furthermore, CMAC uses convergence to avoid the anycast overhead.

VI. CONCLUSION

Existing MAC layer solutions for low duty cycling either consume a lot of energy on periodic synchronization messages or incur high latency due to the lack of synchronization. Thus in this paper we proposes three mechanisms, aggressive RTS, anycast and convergence, to address such problems. We also implement CMAC as the outcome of these three mechanisms above and evaluate it extensively. The experiment and simulation results show that CMAC at low duty cycles can achieve comparable throughput and latency performance as fully awake CSMA protocl, while greatly outperforming other en-

ergy efficient protocols like BMAC, SMAC and GeRaF. Hence, we conclude that CMAC is highly suitable for wireless sensor networks that require low latency and high throughput as well as long network lifetime.

REFERENCES

[1] J. Polastre, J. Hill, and D. Culler, "Versatile Low Power Media Access for Wireless Sensor Networks," in *Proc. SenSys'04*, Nov. 2004, pp. 95–107.

[2] W. Ye, J. Heidemann, and D. Estrin, "Medium Access Control with Coordinated Adaptive Sleeping for Wireless Sensor Networks," *IEEE/ACM Trans. Networking*, vol. 12, no. 3, pp. 493–506, June 2004.

[3] T. van Dam and K. Langendoen, "An Adaptive Energy-Efficient MAC Procotol for Wireless Sensor Networks," in *Proc. SenSys'03*, Nov. 2003, pp. 171–180.

[4] G. Lu, B. Krishnamachari, and C. S. Raghavendra, "An Adaptive Energy-Efficient and Low-Latency MAC for Data Gathering in Wireless Sensor Networks," in *Proc. IPDPS'04*, Apr. 2004, pp. 224–231.

[5] J. Polastre, J. Hui, P. Levis, J. Zhao, D. Culler, S. Shenker, and I. Stoica, "A Unifying Link Abstraction for Wireless Sensor Networks," in *Proc. SenSys'05*, Nov. 2005, pp. 76–89.

[6] "TinyOS," http://www.tinyos.net.

[7] A. Arora, E. Ertin, R. Ramnath, W. Leal, and M. Nesterenko, "Kansei: A High-Fidelity Sensing Testbed," *IEEE Internet Computing*, vol. 10, no. 2, pp. 35–47, Mar. 2006.

[8] "The Network Simulator – ns-2," http://www.isi.edu/nsnam/ns/.

[9] D. S. J. D. Couto, D. Aguayo, J. Bicket, and R. Morris, "A High-Throughput Path Metric for Multi-Hop Wireless Routing," in *Proc. MobiCom'03*, Sept. 2003, pp. 134–146.

[10] J. Padhye, R. Draves, and B. Zill, "Routing in Multi-Radio, Multi-Hop Wireless Mesh Networks," in *Proc. MobiCom'04*, Sept. 2004, pp. 114–128.

[11] K. Seada, M. Zuniga, A. Helmy, and B. Krishnamachari, "Energy-Efficient Forwarding Strategies for Geographic Routing in Lossy Wireless Sensor Networks," in *Proc. SenSys'04*, Nov. 2004, pp. 108–121.

[12] K.-W. Fan, S. Liu, and P. Sinha, "Convergent Anycast: A Low Duty- Cycle MAC Layer for Sensor Networks," Technical Report OSU-CISRC-4/05–TR24, 2005. [Online]. Available: ftp://ftp.cse.ohio-state.edu/pub/tech-report/2005/TR24.pdf

[13] P. Dutta, M. Grimmer, A. Arora, S. Bibyk, and D. Culler, "Design of a Wireless Sensor Network Platform for Detecting Rare, Random, and Ephemeral Events," in *Proc. IPSN'05*, Apr. 2005, pp. 497–502.

[14] "Mica2," http://www.xbow.com/Products/productsdetails.aspx?sid=72.

[15] "CC1000," http://www.chipcon.com/files/CC1000_Data_Sheet_2_2.pdf.

[16] "Stargate," http://platformx.sourceforge.net/home.html.

[17] M. Zorzi and R. R. Rao, "Geographic Random Forwarding (GeRaF) for Ad Hoc and Sensor Networks: Multihop Performance," *IEEE Trans. Mobile Comput.*, vol. 2, no. 4, pp. 337–348, Oct. 2003.

[18] ——, "Geographic Random Forwarding (GeRaF) for Ad Hoc and Sensor Networks: Energy and Latency Performance," *IEEE Trans. Mobile Comput.*, vol. 2, no. 4, pp. 349–365, Oct. 2003.

[19] M. Buettner, G. V. Yee, E. Anderson, and R. Han, "X-MAC: A Short Preamble MAC Protocol for Duty-Cycled Wireless Sensor Networks," in *Proc. SenSys'06*, Nov. 2006, pp. 307–320.

[20] S. Jain and S. R. Das, "Exploiting Path Diversity in the Link Layer in Wireless Ad Hoc Networks," in *Proc. WoWMoM'05*, June 2005, pp. 22–30.

[21] R. R. Choudhury and N. H. Vaidya, "MAC-Layer Anycasting in Ad Hoc Networks," *SIGCOMM Computer Communication Review*, vol. 34, no. 1, pp. 75–80, Jan. 2004.

[22] H. Füßler, J. Widmer, M. Käsemann, M. Mauve, and H. Hartenstein, "Contention-Based Forwarding for Mobile Ad Hoc Networks," *Ad Hoc Networks*, vol. 1, no. 4, pp. 351–369, Nov. 2003.

[23] T. He, B. M. Blum, Q. Cao, J. A. Stankovic, S. H. Son, and T. F. Abdelzaher, "Robust and Timely Communication over Highly Dynamic Sensor Networks," *Real-Time Systems Journal, Special Issue on Real-Time Wireless Sensor Networks*, to appear.

[24] D. Chen, J. Deng, and P. K. Varshney, "On the Forwarding Area of Contention-Based Geographic Forwarding for Ad Hoc and Sensor Networks," in *Proc. SECON'05*, Sept. 2005, pp. 130–141.