Convergent Anycast: A Low Duty-Cycle MAC Layer for Sensor Networks

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

***Abstract*— Energy conservation is critical for extending the lifetime of wireless sensor networks. A simple way to conserve energy is to turn off the radio transceiver for brief durations. Without any synchronization of wakeup schedules, such a mechanism is prone to an increased event detection latency. Current techniques either require periodic synchronization messages to achieve low detection latency using synchronized wakeup, or incur high latency due to the lack of any synchronization. We propose a convergent wakeup protocol, called Convergent-MAC or CMAC, to distributedly determine their wakeup schedules upon event detection. CMAC requires zero communication during quiet periods with no event detections. It converges from anycast-based forwarding in unsynchronized networks to unicast-based forwarding. While converging to the optimal path, nodes distributedly determine their wakeup schedules based only on local information. Simulations show that even when operating at a 1% duty cycle, the throughput and latency for CMAC are comparable to 802.11 based protocols, while using 88.5% lower energy during event detection. For operation at 0.1% duty cycle, CMAC maintains comparable throughput and latency although at the expense of increase in the initial detection latency. Due to its extreme energy conserving capability, CMAC is particularly suited for event triggered sensor networks that often require an extended network lifetime.**

I. Introduction

Extending the lifetime in battery powered wireless sensor networks is critical due to multiple reasons. Frequent human intervention to replace batteries in sensors is an expensive operation especially when sensors are deployed in large numbers or when they are deployed in hard-to-reach areas. Hazardous environments often pose challenges to the deployer if sensors run out of battery too quickly. Sensor networks deployed for detecting chemical leaks, bio-chemical attacks, and poisonous gases in waste management facilities are some such examples. Often sensors show nondeterministic behavior during periods with low battery power, which can result in false positive or false negative detections. For certain applications, deployed sensors are impossible to recover physically. One such example is the Glacsweb project [1] where sensors are embedded 60 feet below the surface of the glaciers for studying their movement patterns. For military and defense applications where sensors are deployed in foreign territories and hostile environments, replacing batteries is often not an option. Thus, it is critical to design protocols for energy conservation in wireless sensor networks.

Sensors networks can be primarily classified into the following two types: Periodic Sampling Networks (PSNs) or Event Triggered Networks (ETNs). PSNs are used for collecting data periodically over a certain time. The frequency of data collection is often predecided. As the network traffic is deterministic, significant pre-deployment planning can optimize the use of network resources. Some such examples are sensors deployed for gathering environmental conditions in forest [2], [3] and sensors for habitat monitoring [4], [5]. ETNs are typically used for detecting non-periodic and infrequent events based on certain triggers. Examples include networks for intrusion detection [6], chemical and biological hazard detection [7], and detection of faults in bridges [8]. ETNs are often more challenging to design as pre-deployment optimizations are difficult to perform due to the non-periodic nature of events. Our study and evaluation focuses on ETNs, although some of the ideas are applicable to PSNs as well.

For a MAC layer protocol for sensor networks, low energy consumption, high throughput, and low delay are essential goals. We seek to design an efficient wakeup schedule and packet forwarding strategy with the following additional specific goals:

- **1:** The protocol must be robust and adaptive to topological changes.
- **2:** The protocol must operate in sparse as well as dense networks.
- **3:** The protocol must operate at very low duty cycles (e.g., 0.1%) during idle times.
- **4:** For sustained traffic, the routes must converge to optimal routes without requiring explicit messaging.
- **5:** The amount of energy consumed during event detection must be optimized.

Several MAC protocols have been proposed to conserve energy in sensor networks. Protocols such as S-MAC [9], [10], T-MAC [11] and DMAC are based on synchronization of wakeup schedules. Frequent synchronization messages consume significant energy in such protocols even when there are no events to report. Other approaches such as GeRaF [12], [13], B-MAC [14], PEAS [15] and GAF [16] do not use synchronization messages. However, GeRaF

is prone to high latency because of unsynchronized sleeping schedules which either leads to multiple packet retransmissions or the discovery of sub-optimal routes. B-MAC requires long preambles leading to high transmission delays and limiting its capability to conserve energy. PEAS and GAF both require overhead messages even when there is no traffic in the network.

To meet our design goals, we propose a combined wakeup schedule and routing approach that converges from anycasting in an unsynchronized network to unicasting on synchronized routes. Due to its capability to converge from anycasting to unicasting, we call our approach Convergent-MAC, or CMAC. Unicasting is based on greedy forwarding like in GPSR [17], and anycasting prefers forwarding nodes that are closer to the greedy choice. Triggered by the flow of data packets, the routes progressively converge to greedy routes at which point the protocol becomes akin to unicasting. Nodes on the unicast route start following a wakeup schedule to efficiently forward data with low latency. The proposed convergent wakeup approach results in low average latency and low energy consumption. It requires zero communication during periods with no event activity.

We implement our protocol in *ns2* [18] and compare it with GeRaF and 802.11 based unicast protocol. The highlights of our simulation based study are as follows:

- For static events scenario, our protocol outperforms IEEE 802.11 based unicast with 100% duty cycle and anycast based GeRaF. With higher throughput and lower latency, CMAC saves more energy than GeRaF. CMAC saves up to 94% energy compared to 802.11 based unicast.
- For dynamic events, CMAC achieves 90% throughput of 802.11 based unicast with comparable latency, while saving up to 88.5% energy.

The rest of the paper is organized as follows. Section II motivates the design of our protocol. Section III presents the details of our protocol. In Section IV, we present results from simulations comparing CMAC with other protocols. Section V summarizes related work on this topic Finally, Section VI concludes the paper.

II. Motivation

Various MAC layer protocols have been proposed for power conservation in sensor networks. In this section, we outline the weaknesses of some of these protocols to motivate the design of CMAC. Our key observations are as follows:

- *Synchronized wakeup is wasteful:* The idea of synchronized wakeup is used in S-MAC [9] and Adaptive S-MAC [10] protocols. In these protocols, communication is restricted to the awake periods of the periodic sleep cycles. To maintain synchronization, nodes need to send periodic synchronization messages. For typical Mica2 hardware, the suggested periodicity of such messages is 10 seconds [9], [10]. Suppose the size of a synchronization packet is 40 bytes, including the physical layer overhead and the preamble. For the Mica2 radio with a raw bit-rate of 38.4Kbps, it takes $8.333ms$ to transmit a synchronization packet. Using $27mA$ as the current-drawn during transmission for Mica2 motes, the energy consumption for a synchronization message is,

$$0.027A \times \frac{40 \times 8}{38400} s \times 3V = 0.675mJ$$

Suppose the network operates at 1% duty cycle by waking up for $0.1s$ every 10 seconds. Using $10mA$ as the current drawn during idle listening for Mica2, the idle energy consumption for a node in every 10 seconds is

$$0.01A \times 0.1s \times 3V = 3mJ$$

Thus, synchronization messages consume almost 18% of the total energy in the absence of data traffic. Eliminating these synchronization messages can increase the network lifetime by 22%. If we further lower the duty-cycle, the synchronization messages will start dominating the total energy consumption. For example, if nodes operate at 0.1% duty cycle, the synchronization messages consume 70% of the total energy. While operating at 0.1% duty cycle, the networks lifetime can be increased by 225% by eliminating the synchronization messages. Thus, for operation at lower duty cycles, the network's lifetime can be significantly increased by eliminating synchronization messages.

- *Anycasting has high overhead in non-synchronized networks:* To address the overhead of synchronization messages in S-MAC and Adaptive S-MAC, GeRaF [12], [13] proposes non-synchronized wakeup for all nodes. GeRaF uses local *anycast* to forward packets to any neighbors closer to the sink. It assumes that the location of neighbors and the location of the sink is known to all nodes. Such information can be made available during network deployment. Anycast based GeRaF has the following three disadvantages over unicast based protocols. First, the computed route could be longer since the optimal forwarding nodes may be asleep during anycasting. Second, the overhead of anycast is higher than unicast as a sending node has to wait to obtain a response from one of the forwarding nodes. The waiting time is typically longer than in unicast. Third, the RTS/CTS packets are usually longer than unicast RTS/CTS as they contain anycast-specific additional information. Thus, although GeRaF does not suffer from the overhead of synchronization messages, it incurs higher overhead during data transmissions.

- *Long preamble leads to high latency for low duty cycles:* As events may be rare in sensor networks, the idle energy is the dominant component of the total energy consumption when synchronization messages are not used. By reducing the duty-cycle, the idle energy consumption could be further reduced at the expense of increase in event detection latency. In B-MAC [14], nodes use a long preamble with fixed length to establish contact with next hop nodes. The preamble must be long enough to ensure that all neighbors can hear it. From [14], the time required to turn on the radio and sample the channel is $3ms$. If the nodes work on 1% duty cycle, nodes will wake up every $300ms$ to monitor the channel. To ensure that all neighbors hear the packet, the preamble in B-MAC must be at least $300ms$ long. We observe that the preamble length increases as the duty-cycle decreases. Moreover, a fixed preamble length is wasteful and leads to higher detection latency, as the node to be contacted may hear the preamble much before 300ms. In addition, if it is an anycast packet as opposed to a unicast packet, the expected length of the preamble will be even shorter.

III. CONVERGENT MAC (CMAC)

The mechanisms in CMAC that address the above goals are as follows:

Based on the above observations, we design the following mechanisms of CMAC to meet the design goals outlined in Section 1.

- *Anycast based initial detection (Goal 1):* CMAC uses zero-communication during periods of inactivity. To address the asynchronous sleep schedules, it uses anycast to establish contact with the forwarding node. Anycast naturally provides robustness against node failures and link quality fluctuations.
- *Aggressive RTS (Goals 2, 3):* For scenarios with low density and low duty-cycle, the possibility of finding an awake forwarding node is also reduced. To discover a forwarding node, anycast is enhanced with multiple RTS transmissions without interim backoffs.
- *Converging from Anycast to Unicast (Goal 4):* Anycast incurs high overhead which is avoided by gradual convergence to unicast for sustained traffic.
- *Staggered Wakeup Schedule (Goal 5):* After convergence, the nodes on the unicast route follow a staggered wakeup schedule. This allows the nodes to synchronize packet transmission and sleep schedules, with immediate upstream and downstream nodes by the packet flow without time synchronization.

The rest of the section presents an overview of CMAC, followed by details on each component of CMAC.

A. CMAC Overview

In an idle network, all nodes use non-synchronized random wakeup schedules with a pre-defined duty cycle, called the *idle duty cycle*. Upon observing an event, nodes use anycasting with aggressive RTS retransmission to establish contact with forwarding nodes and forward packets toward the sink. In anycasting, the nodes with a better metric (in this paper we have assumed remaining distance to the sink as the metric) respond first. The node that receives packets during anycasting remains fully awake for a short duration in anticipation of the next packet. This ensures that at least one node will be awake for subsequent packet's transmission, and anycast will be responded by this node, or by a neighbor with a better metric. Once a node finds a next hop with a high enough metric, it switches from anycasting to unicasting, and only forwards packets to that node in following transmissions. Thus the forwarding path formed by anycast progressively converges to unicast driven by the flow of data packets. Once nodes agree on unicasting, they use synchronized wakeup schedules that are staggered to efficiently forward packets with low latency. After convergence, the wakeup schedule uses a different duty cycle, called the *active duty cycle*, to forward packets. Thus CMAC allows the network to operate at low duty cycles, avoids synchronization messages, and provides high throughput and low latency while ensuring a long network lifetime.

B. Anycast based Initial Detection

Anycasting provides a low latency mechanism to find a next hop node for forwarding packets to. In anycast, the sender attempts to forward the packet to any node among a set of forwarding nodes, with preference given to nodes with a better metric. Like GeRaF [12], [13], we use the remaining distance to the sink as the metric with preference given to nodes closer to the sink. To address problems with low reliability of long links, several other metrics such as ETT [19], ETX [20] and PRRxDist [21], have been proposed. Our anycast protocol can be readily modified to accommodate any such metric.

Our implementation of anycasting uses an RTS-CTS exchange between the sender and the prospective receiver. Other anycast mechanisms [12], [13], [22] can also be used in place of our anycast approach in the CMAC protocol. As there are multiple neighbors who can respond with a CTS, the time instant at which they can start sending CTS is staggered to give preference to neighbors that have better metrics. Following the principles of greedy forwarding [17], only nodes that are closer to the sink than the current sender are allowed to respond with a CTS. We assume that the sensors are aware of their own location, and the sink's location. The area that is closer to the sink than the sender is divided into n regions, R_1, R_2, $\cdots$ R_n such that all nodes in R_i are closer to the sink than nodes in R_j where $i < j$. Figure 1 shows an example where the

area is divided into three regions. Nodes in region R_i send the CTS in time slot i. Nodes that have similar metric (in the same region) will choose a random time between 0 and $k-1$, to decide on a mini-slot to start transmitting the CTS, where k is the number of mini-slots per slot, as shown in Figure 2. The slots are used to give preference to forwarding nodes with better metrics, and the mini-slots are used to avoid collisions between forwarding nodes that have similar metrics (in the same region). If nodes overhear any traffic before transmitting CTS, they cancel the CTS transmission. If neighbors can sense the channel busy when any other neighbor transmits, the incidence of CTS collisions can be further reduced. This requires the range of radio sensing to be at least twice the range of transmission. Nodes in R_2 will have the opportunity to send the CTS only if all nodes in R_1 are sleeping or are unable to send the CTS. Therefore, the nodes closer to the sink always have higher priority in transmitting CTS. The DATA and ACK packets follow the CTS like in the 802.11 protocol [23].

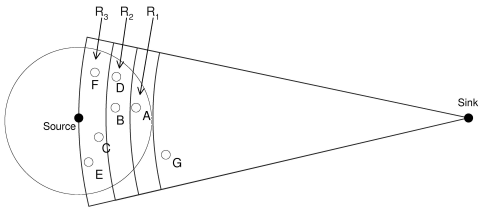

Fig. 1. *Coverage area closer than the sender is divided into regions according to the distance to the sink.*

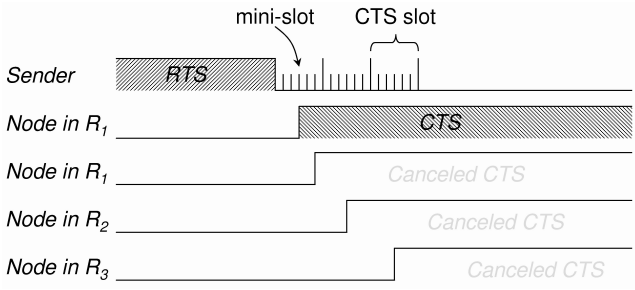

Fig. 2. *Nodes in different regions transmit their CTS in the corresponding CTS slot, while nodes in the same region transmit their CTS in randomly selected mini-slot k.*

C. Aggressive RTS

In order to extend network lifetime, nodes should be able to work on a low duty cycle to reduce idle energy consumption. But for non-synchronized networks, a high density is needed along with low duty cycles, to ensure that at least one forwarding node is awake to receive the packet. To expand the operating region of the protocol and allow operation in low duty cycle mode in sparse networks, B-MAC [14] proposed an approach using fixed length preambles. B-MAC uses a long preamble to ensure that *the selected* receiver will be awake when the sender transmits the packet. However, such a long preamble is not required for our anycast. Consider a scenario with a 1% duty cycle where each cycle is $300ms$. Let L ms be the length of the preamble, and $x = \frac{L}{300}$ be the normalized preamble length. Let the random variable X represent the normalized length of the preamble such that the first forwarding out of n nodes wakes up exactly at the end of the preamble. The CDF (Cumulative Distribution Function) $F(x)$ for X is given by:

$$F(x) = P\{X <= x\} = 1 - (1-x)^n$$

Figure 3 shows the CDF for several different values of n.

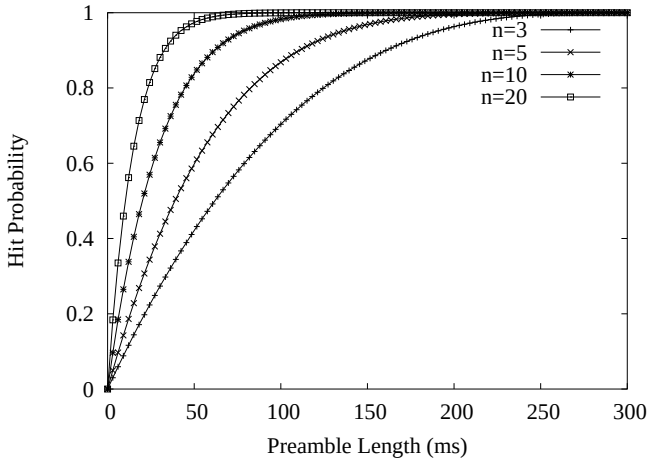

Fig. 3. *The CDF $F(x)$ for 1% duty cycle with a cycle of 300ms.*

The expected value of x is therefore:

$$\begin{aligned} E(x) &= \int_0^1 x \times F'(x)dx \\ &= \int_0^1 xdF(x) \\ &= x \times F(x)|_0^1 - \int_0^1 F(x)dx \\ &= [x \times F(x) - (x + \frac{(1-x)^{n+1}}{n+1})]|_0^1 \\ &= \frac{1}{n+1} \end{aligned}$$

So, the expected preamble length is $\frac{300}{n+1}ms$. Depending on the number of forwarding nodes n, the expected preamble length could be much smaller than $300ms$. A shorter preamble leads to lower event detection latency and lower power consumption. We observe that for anycasting, the preamble length should be a function of the number of forwarding nodes.

Motivated by the above analysis, we propose an *aggressive RTS* mechanism, to wake up neighboring nodes. Unlike the fixed length preamble in B-MAC, aggressive RTS uses multiple RTS packets. When a node wants to transmit a packet, it sends an RTS packet for initiating anycast. If no CTS is received

for the RTS, the next RTS is transmitted without a backoff. The maximum number of such aggressive RTS transmissions is limited by the length of the cycle. The interval between the end of one RTS and the beginning of the next aggressive is the sum of the SIFS, maximum propagation delay, and the length of all CTS slots. CMAC uses a wakeup period of $3ms$, which is higher than this interval to ensure that neighbors do not miss consecutive RTS packets during aggressive RTS transmissions. *Aggressive RTS* is used only when the channel is idle to reduce the contention and interference.

D. Converging from Anycast to Unicast

Although anycast obviates the need for synchronization messages and has better chance of making progress in forwarding packet, it has three main shortcomings. First, anycast does not always choose the best route. The best next hop may not be able to send the CTS because it is sleeping or because of interference. Second, the overhead and delay may be higher. For example, if there are no nodes in R_1, a slot is wasted. Third, anycast packets are usually larger because they contain additional anycast-specific information such as location information or expected forwarding nodes IDs [12], [22].

The forwarding node that responds with a CTS may have a metric close to the best metric. For example a node in region R_1 (Figure 1) may respond to an RTS. When such a case occurs, there is no need to perform anycast for the subsequent packets as this node can be chosen as the next hop for all such packets. Subsequent packets can simply use unicast. The sender indicates its willingness to synchronize its schedule with the receiver by using a synchronization flag in the DATA packet header. The MAC layer ACK from the receiver serves as a confirmation for the receiver's acceptance of a synchronized staggered schedule. After converging to unicast, the two nodes start following a staggered wakeup schedule with the active duty cycle. The sender and receiver will maintain the schedule as long as there is traffic between them. If there is no traffic for a certain period, which may be due to the end of the event or the failure of the sender or receiver, the synchronization times out, and the nodes go back to the non-synchronized mode and start following the idle duty cycle.

The node that is selected as the receiver during anycasting remains fully awake for a fixed duration if it is not selected to synchronize with the sender. This ensures that anycast for the next packet will be received by the current receiver or a neighbor that has better metrics than the current receiver, and also reduces the overhead incurred by *aggressive RTS*. If it cannot find a better next hop to synchronize with for a duration, it assumes that there is no better node and starts to synchronize with the latest receiver. Thus anycasting progressively converges to unicasting on a synchronized route, as shown in Figure 4.

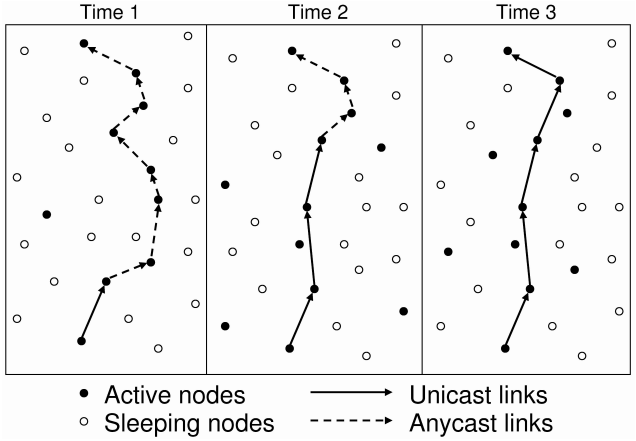

Fig. 4. *Anycasting route converges to unicast route gradually.*

Notice that the packets can still be forwarded to the sink using anycasting before the route converges. The converging mechanism only tries to find a better node as next hop, and the convergence occurs gradually while the data is forwarded to the sink. Therefore before the route converges, the packets are still forwarded to the sink, although with a higher overhead due to anycasting on some hops.

E. Synchronization on a Single Route

After nodes converge from anycast to unicast, they use a synchronized schedule with the *active duty-cycle* to forward packets. A 100% active duty-cycle results in high contention between upstream and downstream nodes, and low channel utilization. To reduce such contention, the transmission and reception slots are split. The synchronized schedule is illustrated in Figure 5. Each wakeup cycle has a sleep time slot and an active time slot. In active time slot, time is divided into a receiving slot and a transmitting slot. Nodes can only transmit data during the transmitting slot. Therefore, the downstream nodes must schedule their receiving time slot to match their upstream node's transmitting slot. Figure 6 illustrates the synchronization schedules of a few synchronized nodes. The receiving and transmitting slots are staggered such that the upstream nodes can transmit to downstream nodes without contention between them. The staggered schedule allows nodes to forward packets from the source to the sink with low delay.

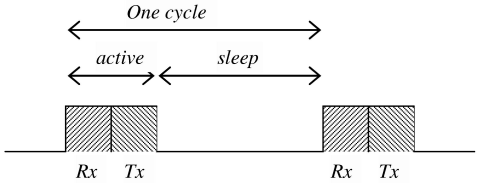

Fig. 5. *Synchronization Schedule.*

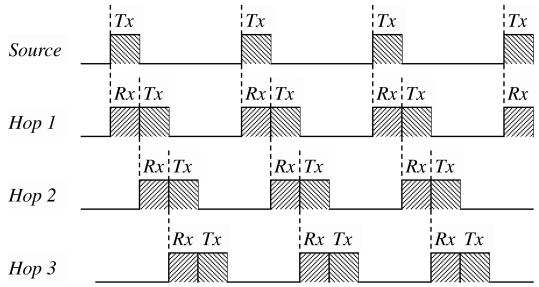

Fig. 6. *Staggered synchronization schedules.*

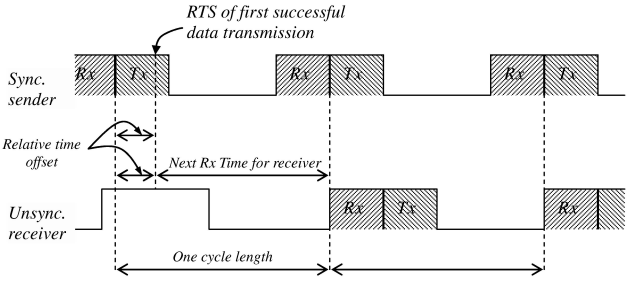

Fig. 8. *A synchronized sender synchronizes with an unsynchronized receiver. The sender explicitly indicates the receiver the time offset of its schedule in DATA header.*

When two nodes agree to synchronize, they must schedule their wakeup periods in a staggered way. We consider the following two cases for synchronization.

- *The sender is the source and is not synchronized.* When a sender is not synchronized, the schedule can be started at any time. Figure 7 illustrates a scenario where an unsynchronized sender wants to synchronize with the receiver. The schedule will be started as of the time the sender sends the RTS packet for the first successful data transmission.
- *The sender is an intermediate node, and is already synchronized with its upstream node.* As the sender is already synchronized, it can not change its schedule. So when it needs to synchronize, it explicitly indicates the time elapsed since the beginning of the current transmitting slot (see Figure 8). The receiver uses this offset to determine its staggered wakeup schedule to properly match the sender. Therefore even if the transmission failed for the first few tries, the receiver can still know the time to start the schedule.

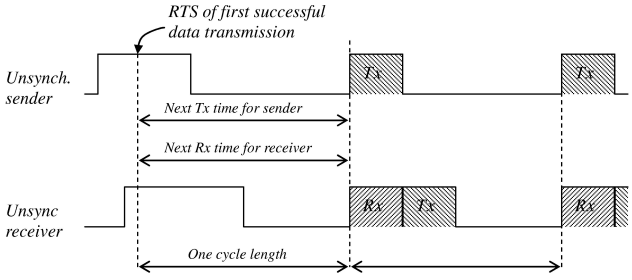

Fig. 7. *An unsynchronized sender synchronizes with an unsynchronized receiver. The schedule will be started as of the time the sender sends the RTS packet of the first successful data transmission.*

F. Synchronization on Merging Routes

Multiple sources may simultaneously send data in case of a static event that triggers multiple nodes or in case of a mobile event. When multiple sources need to report data to the sink synchronization needs to be managed across merging routes. If a sender is not synchronized but the receiver is already synchronized with another sender (at the junction of two merging flows), it follows the receiver's schedule. The receiver indicates the time elapsed since the beginning of the last receiving time slot in the CTS header, and the sender learns when to start its transmitting slot. If both the sender and the receiver are synchronized with their upstream nodes (receiver is synchronized with another sender) but they are not synchronized with each other, the following approaches can be used to adjust their synchronization:

- The sender can match the receiver's schedule and request its upstream nodes to adjust their wakeup schedules. However, this causes a ripple effect that needs to be propagated to the leaf nodes of the tree.
- The receiver switches to a wakeup schedule that satisfies the new sender as well as the old sender(s). However, this requires the receiver to maintain a higher active duty cycle (see Figure 9).
- The sender splits its receiving and transmitting slots to match with its upstream as well as downstream nodes as shown in Figure 10. We have chosen this approach to implement as it incurs lower overhead compared to the other approaches.

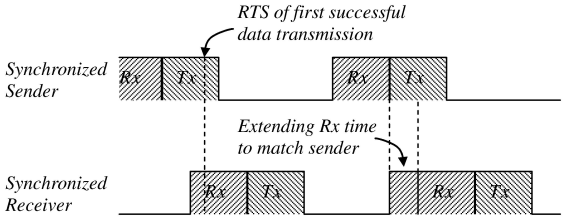

Fig. 9. *A synchronized receiver accommodates a synchronized sender by extending its receiving slot.*

G. CMAC Extension

Although staggered wakeup alleviates the contention between upstream and downstream nodes, multiple merging flows can cause significant cross-route interference. Such interference is observed when multiple events occur simultaneously and also when

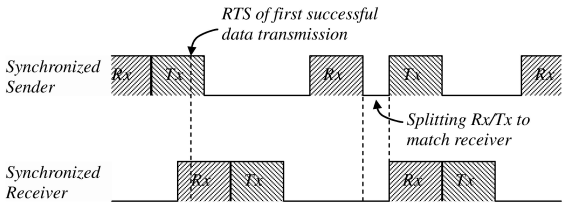

Fig. 10. *A synchronized sender accommodates a synchronized receiver by splitting its receiving and transmitting slots to match the receiver's slots.*

the event is mobile. To address this problem, we propose an extension to CMAC. After converging from anycast to unicast, nodes remain fully awake (rather than following an active duty cycle) without any synchronization. So, nodes can transmit and receive at any time rather than wait for the Rx or Tx slots. Although it increases the energy consumption during active periods, only the nodes that forward data are required to remain fully awake. When traffic ceases, the nodes go back to follow the non-synchronized idle duty cycle. We study the trade-offs between CMAC and CMAC Extension as part of our performance evaluation.

IV. Performance Evaluation

To evaluate the proposed scheme, we compare the throughput, latency and normalized energy consumption of our protocols with other protocols using the network simulator *ns2*. Our study is based on the following four protocols:

- **802.11:** Using RTS-CTS-DATA-ACK 802.11 protocol with optimal routing. Nodes have a duty cycle of 100%. This protocol servers as the baseline for optimal throughput performance.
- **GeRaF:** Using RTS-CTS-DATA-ACK anycasting protocol described in Section III-B. Our implementation has some differences from the description of GeRaF in [12]. First we do not use busy tone to detect if the channel is busy. Second, we use the anycast protocol described in III-B, which is slightly different from the anycast protocol described in GeRaF [12].
- **CMAC:** Our proposed scheme described in Section III. Nodes work on a 1% idle duty cycle.
- **CMAC-Ext:** Our proposed scheme with the extension described in Section III-G. When nodes are synchronized, they remain active with a 100% duty cycle to receive and forward packets.

We use $250m$ as the transmission range, but our protocol works for any radio transmission range. We configure the bandwidth to 38.4Kbps, the maximum transmission power to 27mA, the receiving power to 10mA and idle listening power to 10mA. The CTS for anycast contains extra information of the receiver's address (6 bytes) and the offset value of the receiver's schedule (2 bytes) as described in III-F, therefore the size of CTS for anycast is 14+6+2 = 22 bytes. The length of a time slot for receivers to reply with a CTS should be long enough such that the probability of CTS collision for nodes in the same region is low. We use 0.2ms in the following simulations. Table IV lists the parameters we used in the simulations.

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
Preamble+PLCP	24 bytes	Anycast CTS	22 bytes

TABLE I
SIMULATION PARAMETERS

In this section, we first present results on static event scenario followed by evaluating the performance in mobile event scenarios for varying parameters including, data rate, initial duty-cycle, node density, location error, and the link quality.

A. Static Event

First we evaluate the performance in a simple scenario with one static event to justify our design. The performance is shown in Figures 11, 12, and 13. We can see that in this scenario, CMAC and CMAC-Ext have the same throughput and delay as 802.11 and consume only 6% energy as 802.11 does when the date rate is less than 4 packets/s. This is because the channel is not full loaded. While in high data rate, the CMAC protocol outperforms all others protocols in terms of throughput, delay and normalized energy consumption. That is because by splitting the RX and TX time slots, nodes can avoid contention between upstream and downstream nodes, therefore packets can be quickly forwarded to the sink. These graphs do not show GeRaF performance as only a few packets or no packets can be transferred to the sink successfully using GeRaF.

The rest of the evaluation is based on mobile events in the grid network in different scenarios. Unless otherwise mentioned, the configuration for simulations is as follows. An event moves randomly in the grid network for 400 seconds at the speed of 10 m/s. The grid size is 20 nodes by 20 nodes, and the distance between two nodes is 100m. The sensing range of sensors is configured such that the entire region is covered with the least value. Therefore the sensing range is 71m if the distance between two nodes is 100m ($100 \times \sqrt{2}/2 \approx 71$). In the following simulation, we use 1% idle duty cycle in our protocols and in GeRaF unless otherwise mentioned. We did not consider variation of the event size in this paper due to space limitation.

B. Data Rate

Figures 14, 15, and 16 show the simulation results of throughput, latency and energy consumption of different protocols for different date reporting rates. In

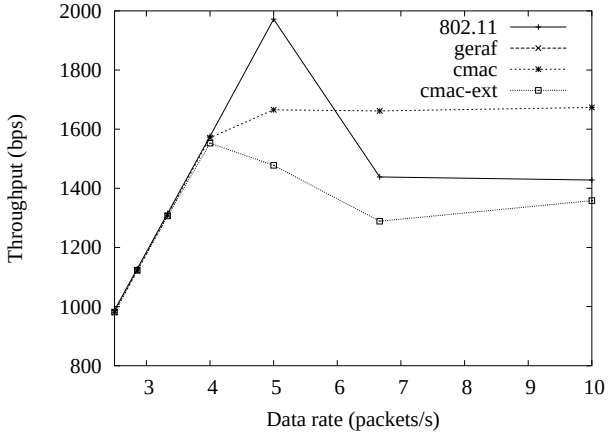

Fig. 11. *Average throughput of protocols with one static event. Using only 1% idle duty cycle, CMAC outperforms other protocols because it separates RX and TX slots. It reduces contention and increases the channel utilization.*

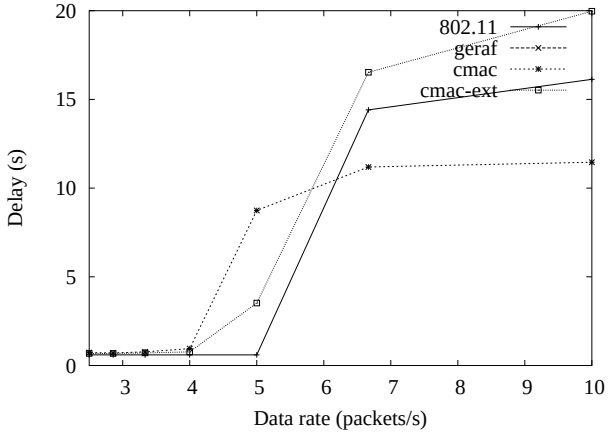

Fig. 12. *Average end-to-end latency delay with one static event.*

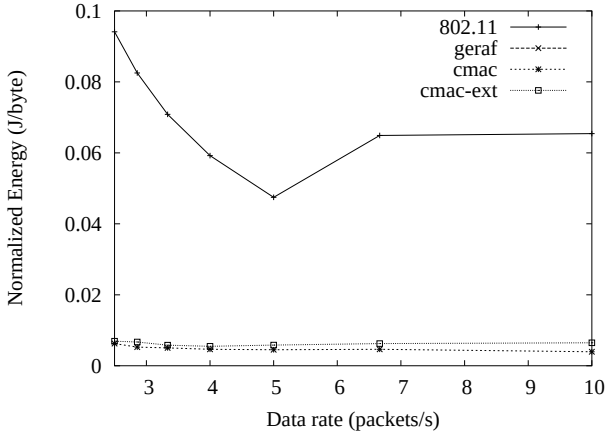

Fig. 13. *Normalized energy consumption, the amount of energy spent per byte. CMAC consumes only 6% energy in comparison to 802.11, and consumes 60% of the energy consumed by CMAC-Ext.*

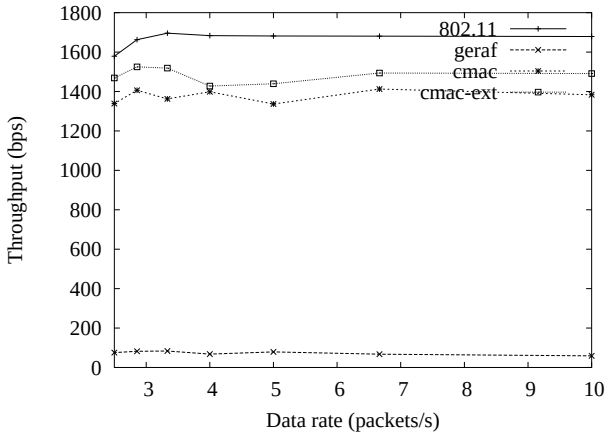

Fig. 14. *Average throughput of protocols with one mobile event. Using only 1% idle duty cycle, CMAC and CMAC-Ext can achieve 80% to 90% throughput of 802.11.*

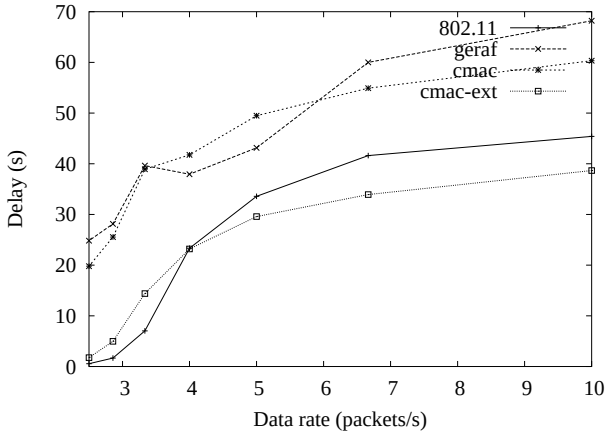

Fig. 15. *Average end-to-end latency delay with one mobile event.*

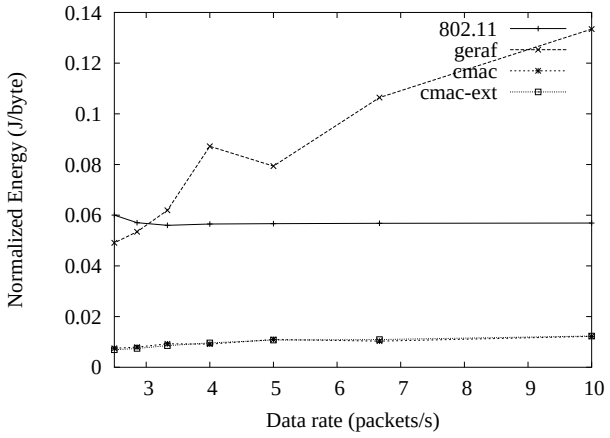

Fig. 16. *Normalized energy consumption, the amount of energy spent per byte.*

mobile event scenarios, CMAC and CMAC-Ext still use the least energy, while achieving 80% to 90% of throughput in 802.11. The reason that the throughput of CMAC and CMAC-Ext are lower is due to two reasons. First, the duty cycle is only 1% initially, which limits the initial throughput. Second, the event may move out of a sensor's sensing range before the route fully converges. However, 802.11 achieves higher throughput with very high power consumption as nodes remain 100% awake. When the network is idle, 802.11 will spend 100 times more energy than CMAC/CMAC-Ext. When there are events, 802.11 provides only 10% higher throughput than CMAC-Ext, but spends 5 to 9.5 times more energy than CMAC/CMAC-Ext. Therefore CMAC/CMAC-Ext are more suitable for networks that require longer network lifetime.

In mobile scenarios, CMAC-Ext has better performance than CMAC. That is because in such scenarios, there are multiple neighboring nodes transmitting concurrently. The splitting of the RX and TX slot in CMAC limits the channel utilization under such high interference.

The latency for CMAC-Ext is lower than 802.11. The reason is that the throughput in CMAC-Ext is lower than 802.11, therefore more packets are dropped. The dropped packets are usually those packets with more retransmissions and higher delay. Therefore the average delay in CMAC-Ext is lower than 802.11.

C. Initial Duty Cycle

Now we vary the idle duty cycle from 0.1% to 1% to evaluate the impact of the idle duty cycle on our protocols. If the protocol can work on very low duty cycle, the network can extend its lifetime further. Figures 17, 18, and 19 show the performance on different idle duty cycle using data rate of 10 packets/s.

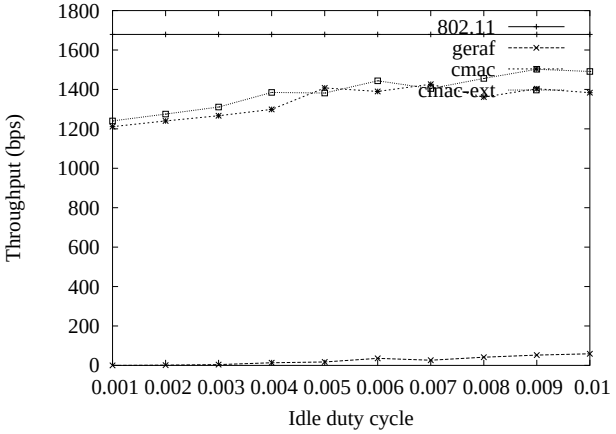

Fig. 17. *Average throughput v.s. idle duty cycle.*

We can see that the throughput decreases gradually when the initial duty cycle becomes lower. The throughput drops about 17% when duty cycle changes from 1% to 0.1%, and the normalized energy consumption dropped about 27%. The average throughput and delay are close and energy consumption is less when nodes work on 0.1% duty cycle. In an idle network operating at 0.1% duty-cycle extends the network lifetime by 10 times in comparison to a 1% duty cycle. However, this energy saving comes at the expense of higher initial detection latency. Tracing the simulation results data we found that the delay for the first packet to arrive at the sink increases from 2 seconds to almost 14 seconds. The tolerance to such delays are dependent on the application's needs.

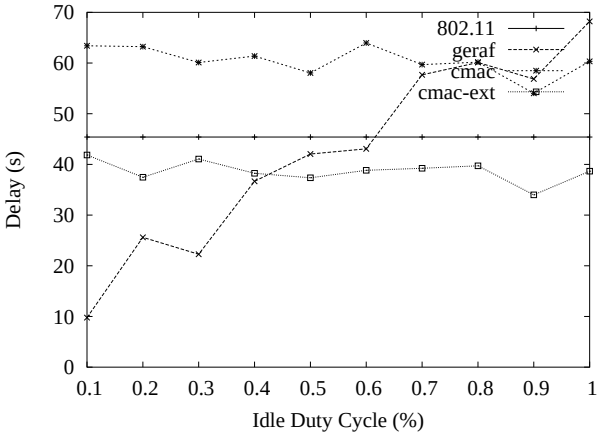

Fig. 18. *Average end-to-end delay v.s. idle duty cycle.*

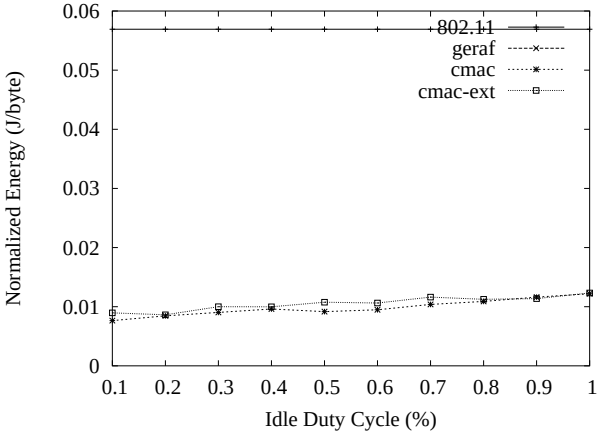

Fig. 19. *Normalized energy consumption v.s. idle duty cycle.*

D. Node Density

Next we evaluate the performance of our protocol on network in different node density.

We can see that in our protocols, the throughput, delay and normalized energy increase gradually when the node density becomes higher. This is because with more nodes, we have more choices for routes, therefor we can have higher throughput. The delay is lower because the number of packet transmitted to the sink is smaller. And with more nodes, the network consumes

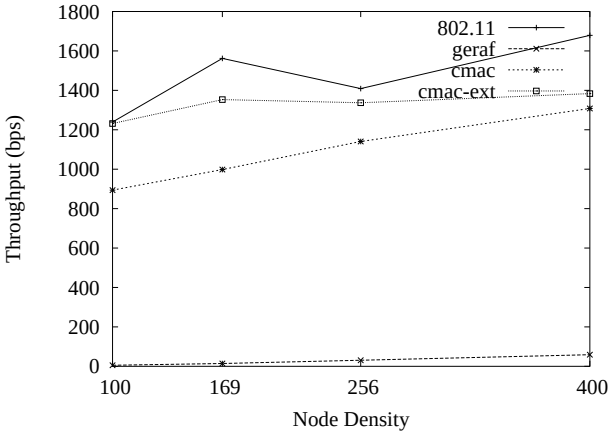

Fig. 20. *Average throughput v.s. Node density.*

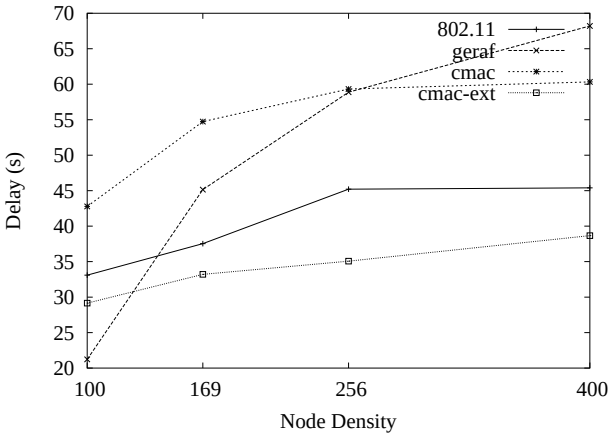

Fig. 21. *Average end-to-end delay v.s. Node density.*

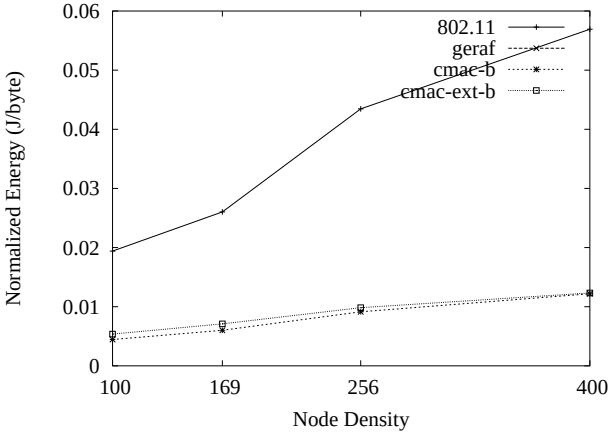

Fig. 22. *Normalized energy consumption v.s. Node Density. The normalized energy consumption of GeRaF is so high because the number of received packet is too low, its energy consumption is not shown here.*

more energy on idle listening. The initial detection delay increases a little, from 2 to 3 seconds. This shows that the node density has only little impact on our protocols.

E. Location Error

Our protocols rely on an anycast protocol to find a feasible route, and most of the anycast protocols depend on the nodes' location to decide the forwarding nodes. In realistic environment, the location information may not be absolutely accurate, therefore we evaluate the impact of inaccuracy on our protocols.

We conducted simulations with varied errors of the location, from 0 to $50m$. The minimum internodal separation in all the experiments was $100m$. Because of the limited space, we did not put the graphs here. The simulation result shows that the location inaccuracy has no impact on our protocols because as long as the anycast protocol can find a route, our protocols can forward packets to the sink.

F. Link Quality

Last we evaluate the impact of the link quality on our protocols by varying the shadowing deviation σ in $ns2$'s propagation shadow model. σ represents the variation of the received power at certain distance, which represents the link quality. Figures 23, 24, and 25 shows the results.

We can see that when σ increases, the performance of all protocols drops since the link quality is getting worse. Our protocols do not suffer more than 802.11 in terms of throughput, and CMAC-Ext still performs better than 802.11 in terms of delay and energy consumption.

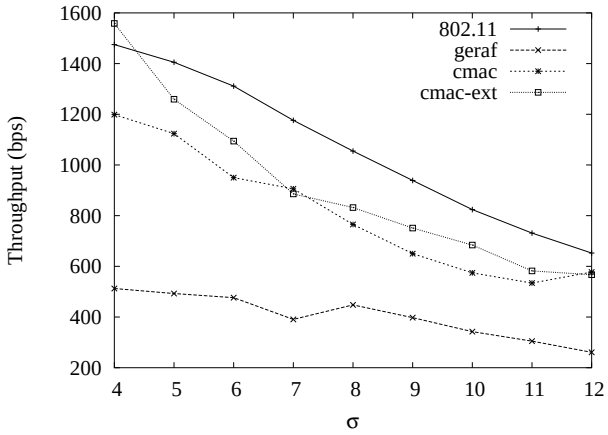

Fig. 23. *Average throughput v.s. σ.*

V. Related Work

The life time of wireless sensor networks can be increased by putting nodes into sleep mode for brief periods periodically. However, nodes are unable to forward data while they are sleeping. Therefore different approaches are proposed to ensure that packets can be forwarded to the destination despite that

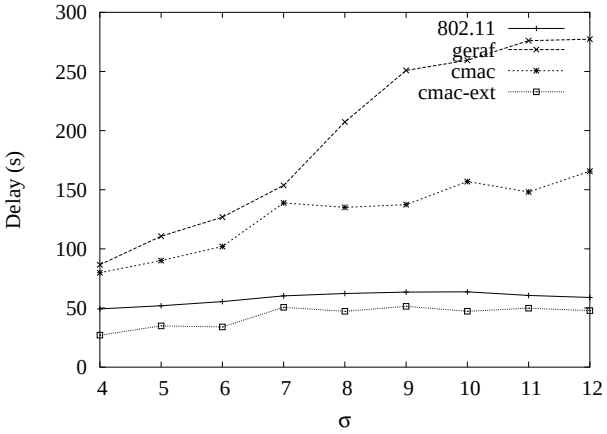

Fig. 24. *Average end-to-end delay v.s. σ.*

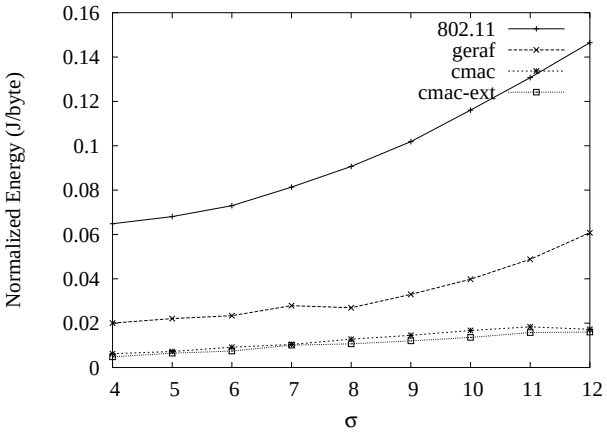

Fig. 25. *Normalized energy consumption v.s. σ.*

some nodes are sleeping. These approaches can be broadly divided into two categories: synchronized and unsynchronized.

Synchronized approaches: Protocols using this mechanism require nodes to periodically synchronize with their neighbors using synchronization messages, and nodes wake up and sleep according to the synchronized schedule.

In S-MAC [9], nodes exchange their wakeup and sleep schedules before following common schedules among neighborhood nodes. Thus nodes can work at low duty cycles. In later work [10], the authors extend S-MAC with adaptive listening using overhearing during listening period to reduce the latency.

T-MAC [11] uses the same mechanism as S-MAC to synchronize nodes, but save more energy by ending the listening period dynamically to reduce the amount of idle listening. In S-MAC, the listening time is fixed. When a node wakes up to listen, it will remain active until its listening time ends. In T-MAC, instead of idly listening the channel for entire listening time, nodes will go back to sleep if it does not hear anything withing a timeout.

DMAC [24] and [25] schedule the nodes' active/sleep time like a ladder from sources to the sink such that the packet can be forward to the sink without delay. The active time is divided into receiving and transmitting slots to avoid interference with the upstream and downstream nodes. In DMAC, nodes can dynamically adapt to higher traffic load by using more-to-send (MTS) packet to adjust their wakeup frequency.

Unsynchronized approaches: Synchronization messages consume significant energy in networks even if there is no event. Therefore other approaches are proposed to avoid the overhead incurred by synchronization messages.

In [26], nodes wakes up after getting triggered by communication events and work at 100% duty cycle while being active. GAF [16] divides the network into virtual grids and maintains one awake node in each grid cell to ensure connectivity. PEAS [15] provides a resilient, long-lived sensor network for an unreliable environment by explicit querying and response when a node wakes up.

In B-MAC [14], nodes wake up periodically to check if there is traffic. It uses clear channel access (CCA) to detect if the channel is busy, and use low power listening (LPL) to check the radio activity. When a node wakes up, it uses CCA to check the radio activity. If no activity is detected, the node goes back to sleep. If activity is detected, the node stays awake to receive the packets. In order to let the nodes detect the traffic reliably, the packet preamble length must be long enough to be detected. For example, if a node checks the channel every 100ms, the preamble must be at least 100ms long to be detected. By using low power listening, nodes can work on very low duty cycle and therefore can save energy and extend network lifetime.

GeRaF [12][13] eliminates the synchronization and probing messages by using anycast. In GeRaF, nodes know the location of their neighbors and the sink. When a node needs to send a packet, it broadcasts an RTS to all its neighbors. Nodes that receive the RTS reply with a CTS packet according to their distance to the sink. Nodes that are closer to the sink will first send the CTS. When the source node receives the CTS, it sends the DATA packet to the node answering the RTS. Other potential receiving nodes overhearing the CTS or the data packet cancel their CTS transmission. This mechanism is simple because nodes do not have to maintain synchronization or exchange schedule information with their neighbors. Moreover, with high enough node density, nodes can work on very low duty cycle while maintaining network connectivity. [27] proposes another anycast protocol in which transmitters specify the potential receiver list along with their replying priorities in a modified RTS packet.

VI. Conclusion

This paper proposes CMAC, a MAC layer approach for maximizing network lifetime and maintaining high throughput and low detection delay for event triggered sensor networks. During idle periods, the nodes are unsynchronized and use a low duty cycle (termed idle duty cycle) to randomly wakeup and sleep. CMAC initially uses anycasting due to lack of synchronization across nodes, but converges to unicasting to reduce latency. The converged routes switch to using an active duty cycle with staggered synchronized schedules. The staggered wakeup schedules result in routes with less interference and low latency. The use of low idle duty cycle during idle periods and active duty cycle during active periods result in low energy consumption and longer network lifetime. A simple extension to CMAC, termed CMAC-Ext is proposed for dealing with cross-route interference. In CMAC-Ext nodes remain fully awake as long as they are forwarding traffic and time is not split into transmission and reception slots. Using extensive simulations, we observe that for detection of static events, CMAC outperforms other protocols in terms of throughput and latency with only 6% energy consumption of 802.11 unicast. For mobile events, CMAC-Ext achieves 90% throughput of 802.11 unicast with similar latency while spending only 11% to 22% energy of unicast. With higher tolerance for initial detection latency, CMAC/CMAC-Ext can even work at 0.1% duty cycle with comparable performance with 802.11. Based on our study, we conclude that CMAC is highly suited for event triggered sensor networks that require long network lifetime. As part of future work, we plan to implement CMAC on the Mica2 motes to evaluate its performance.

References

[1] " GlacsWeb: Autonomous Sub-glacial Probes," http://envisense.org/glacsweb.htm.

[2] "Networked Infomechanical Systems," http://www.cens.ucla.edu .

[3] "Center for Embedded Networked Sensing at UCLA," http://www.cens.ucla.edu .

[4] J. Polastre, "Design and Implementation of Wireless Sensor Networks for Habitat Monitoring," Master's Thesis, University of California at Berkeley, Spring 2003.

[5] A. Mainwaring, R. Szewczyk, J. Anderson, and J. Polastre, "Habitat Monitoring on Great Duck Island," http://www.greatduckisland.net.

[6] A. Arora, P. Dutta, and S. Bapat, "Line in the Sand: A Wireless Sensor Network for Target Detection, Classification, and Tracking," OSU-CISRC-12/03-TR71, 2003.

[7] S. Corporation, "Chemical/Bio Defense and Sensor Networks," http://www.sentel.com/html/chemicalbio.html .

[8] D. Culler, J. Demmel, G. Fenves, S. Kim, T. Oberheim, and S. Pakzad, "Structural Health Monitoring of the Golden Gate Bridge," http://envisense.org/glacsweb.htm.

[9] W. Ye, J. Heidemann, and D. Estrin, "An Energy-Efficient MAC Protocol for Wireless Sensor Networks," in *INFOCOM 2002*, 2002.

[10] W. Ye, J. Heidemann, and D. Estrin, "Medium access control with coordinated adaptive sleeping for wireless sensor networks," in *IEEE/ACM Tran. on Networking, Volume 12, Issue 3 (June 2004)*, 2004.

[11] T. van Dam and K. Langendoen, "An adaptive energy-efficient MAC protocol for wireless sensor networks," in *Proceedings of the 1st international conference on Embedded networked sensor systems*, 2003.

[12] M. Zorzi and R. R. Rao, "Geographic Random Forwarding (GeRaF) for Ad Hoc and Sensor Networks: Energy and Latency Performance," in *IEEE Trans. on Mobile Computing, October-December 2003(Vol. 2, No. 4)*, 2003.

[13] M. Zorzi and R. R. Rao, "Geographic Random Forwarding (GeRaF) for Ad Hoc and Sensor Networks: Multihop Performance," in *IEEE Trans. on Mobile Computing, October-December 2003(Vol. 2, No. 4)*, 2003.

[14] J. Polastre, J. Hill, and D. Culler, "Versatile low power media access for wireless sensor networks," in *Proceedings of the 2nd international conference on Embedded networked sensor systems*, 2004, pp. 95 – 107.

[15] F. Ye, G. Zhong, S. Lu, and L. Zhang, "PEAS: A Robust Energy Conserving Protocol for Long-lived Sensor Networks," 2003.

[16] Y. Xu, J. S. Heidemann, and D. Estrin, "Geography-informed energy conservation for Ad Hoc routing," in *Mobile Computing and Networking*, 2001, pp. 70–84.

[17] B. Karp and H. T. Kung, "GPSR: Greedy Perimeter Stateless Routing for Wireless Networks," in *Proc. of ACM MOBICOM 2000*, Aug. 2000, pp. 243–254.

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

[19] J. Padhye, R. Draves, and B. Zill, "Routing in multi-radio, multi-hop wireless mesh networks," in *MOBICOM 2004*, 2004.

[20] D. D. Couto, D. Aguayo, J. Bicket, and R. Morris, "A High-Throughput Path Metric for Multi-Hop Wireless Routing," in *MOBICOM 2003*, 2003.

[21] K. Seada, M. Zuniga, A. Helmy, and B. Krishnamachari, "Energy-Efficient Forwarding Strategies for Geographic Routing in Lossy Wireless Sensor Networks," in *SENSYS 2004*, 2004.

[22] R. R. Choudhury and N. H. Vaidya, "MAC-Layer Anycasting in Wireless Ad hoc Networks," in *UIUC ECE Technical Report*, 2003.

[23] IEEE, "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications," ISO/IEC 802-11:1999, 1999.

[24] G. Lu, B. Krishnamachari, and C. S. Raghavendra, "An Adaptive Energy-Efficient and Low-Latency MAC for Data Gathering in Wireless Sensor Networks," in *18th International Parallel and Distributed Processing Symposium (IPDPS'04)*, 2004.

[25] M. L. Sichitiu, "Cross-layer scheduling for power efficiency in wireless sensor networks," in *IEEE INFOCOM 2004, vol. 23, no. 1*, 2004, pp. 1741–1751.

[26] R. Zheng and R. Kravets, "On-demand Power Management for Ad Hoc Networks," in *Proc. IEEE INFOCOM '03*, 2003.

[27] S. Jain and S. R. Das, "Exploiting Path Diversity in the Link Layer in Wireless Ad Hoc Networks," http://www.cs.sunysb.edu/ samir/Pubs/anycast.pdf, 2004.