Data-flow Driven Convergent Wakeup Structures in Event Triggered Sensor Networks

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

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 structure that dynamically converges from performing anycast-based to unicast-based data forwarding. Convergence is triggered by the flow of data packets. It requires zero communication during quiet periods with no event detections. While converging to the optimized path, nodes distributedly determine their wakeup schedules based only on local information. Quick convergence leads to low event reporting latency. Simulations comparing with a 802.11 based MAC for typical sensor densities show that even when operating at a 10% duty cycle, the throughput and latency of our protocol outperforms 802.11 based protocols while saving 88.5% energy in static event scenarios. With an event moving at a speed of 10 m/s in a network operating at a 10% duty cycle, the throughput and latency of our protocol remains close to 802.11 based protocols. Due to its heavy energy saving capability, our protocol is particularly suited for mostly sleeping networks that often require an extended network lifetime.

1 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 indeterministic 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 [?] 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 use of network resources. Some such examples are sensors deployed for gathering environmental conditions in forest [?, ?] and sensors for habitat monitoring [?, ?]. ETNs are typically used for detection of non-periodic and infrequent events based on certain triggers. Examples include networks for intrusion detection [?], chemical and biological hazard detection [?], and detection of faults in bridges [?]. 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 for PSNs as well.

Research on ultra low power hardware design [?] addresses power conservation at the physical layer of the network protocol stack. Such research is complementary to our work. Renewable and alternate sources of energy such as solar energy and fuel cells, can be used to solve the energy needs of sensor networks. However, the required size of solar panels and the dependency on availability of sunlight limit the applicability of solar energy solutions to most environments. Technologies such as fuel-cells are in their infancy, and are prohibitively expensive for large sensor networks.

We seek to design a wakeup schedule and packet forwarding strategy with the following three goals. First, energy consumption should be minimized during occurrence of events as well as during periods with no events. Second, events must be detected with low latency. Third, the throughput should be comparable to the scenario with 100% duty cycle. Designing solutions to meet the above goals is challenging due to the seemingly contradictory nature of some of the goals. Minimizing energy consumption during inactive periods needs minimal or no communication which makes it difficult for nodes to stay synchronized to a fixed sleeping pattern.

Several MAC protocols have been proposed to conserve energy in sensor networks. Protocols such as S-MAC [?, ?], T-MAC [?] and DMAC are based on synchronization of wakeup schedules. Frequent synchronization messages consume significant energy in such networks even when there are no events to report. Other approaches such as GeRaF [?, ?], B-MAC [?], PEAS [?] and GAF [?] do not use synchronization messages. However, GeRaF is prone to high latency because of non-synchronized sleeping schedules which either leads to multiple packet retransmissions or the discovery of sub-optimal routes. B-MAC requires long preambles and brief sleeping periods that limit 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 [?], 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 structure 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* [?] and compare it with GeRaF, 802.11 based unicasting, and anycasting approaches. 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, we can save more energy than GeRaF, up to 88.5% energy spent in 802.11 based unicast.

- For dynamic events, our protocol achieves 90%∼99% throughput of 802.11 based unicast with comparable latency, while saving up to 72%∼83% energy spent in 802.11 based unicast.

The rest of the paper is organized as follows. Section 2 motivates the design of our protocol. Section 3 presents the details of our protocol. In Section 4, we present results from simulations comparing CMAC with other protocols. Section 5 summarizes related work on this topic Finally, Section 6 concludes the paper.

2 Motivation

Various MAC layer protocols have been proposed for power conservation in sensor networks. In this section, we first outline the key weaknesses of some of these protocols to motivate our design. We then present a simulation result comparing our protocol with other approaches in a simple scenario to illustrate the advantages of our scheme. We make the following key observations based on the properties of the current MAC protocols for power conservation.

- *Synchronized wakeup is wasteful:* If nodes in a network converge to a synchronized wakeup schedule, communication can be restricted to the awake periods to conserve energy. Such synchronization often requires periodic messages. In [?, ?], authors reported that for typical sensor hardware, one message every 10 seconds is needed for synchronization. Protocols such as S-MAC [?, ?] and T-MAC [?] are based on synchronized wakeup schedule. Although the latency is low in such protocols, the overhead of synchronization is high.

  The DMAC [?] protocol requires the overhead of synchronization, but unlike S-MAC, it staggers the wakeup schedule of nodes to reduce the latency even further. Although staggering of wakeup schedule of nodes is a desired feature for low latency, the overhead of synchronization messages is not.

- *Anycasting has high latency in non-synchronized networks:* To address the overhead of synchronization in resource constrained sensor networks, protocols for networks with non-synchronized wakeup schedule have been proposed. GeRaF [?, ?] is one such protocol that uses local anycast to forward packets to any one of the neighboring nodes closer to the destination. 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 and can be maintained with very low periodic updating mechanisms. Anycast based GeRaF has the following two disadvantages over unicast based protocols. First, the route computed could be longer as the optimal forwarding nodes may be sleeping. Second, the overhead of anycast is higher than unicast as a sending node has to wait to hear a response from one of the forwarding nodes, and the wait is typically longer than in unicast. In GeRaF there is no synchronization overhead, but the latency is typically higher than synchronized protocols.

Based on the above observations, we derive the following four design principles for our protocol. First, synchronized wakeup should not be used, and anycasting can be used during initial event detection. Second, for sustained traffic, the forwarding mechanism on the routes should progressively converge from anycast to unicast, to avoid the above mentioned downsides of anycast. Third, once the routes have converged, the forwarding nodes can use a staggered pattern of wakeup schedule like DMAC to forward packets with low latency but without requiring a 100% duty cycle even while forwarding packets. And fourth, convergence to unicasting and the staggered wakeup schedule can be driven by the flow of data packets.

2.1 An Example

Our design of the protocol, called Convergent MAC or CMAC, is based on the above principles. To demonstrate its performance, we consider a linear topology of 10 nodes, with 200 m separation between adjacent nodes. The transmission range is 250 m. The traffic is from a single sender at one end to a single sink at the other end. We compare our protocol to unicast based 802.11 with 100% duty cycle and anycast based forwarding with 100% duty cycle. For simplicity we assume that the initial duty cycle is also 100% for all three protocols. We consider a traffic load of 10 packets per second. Figure 1 shows that the latency is significantly less while the throughput is higher than the other two protocols. The reduction in interference due to the staggered wakeup schedule after convergence has resulted in improvement in both latency and throughput.

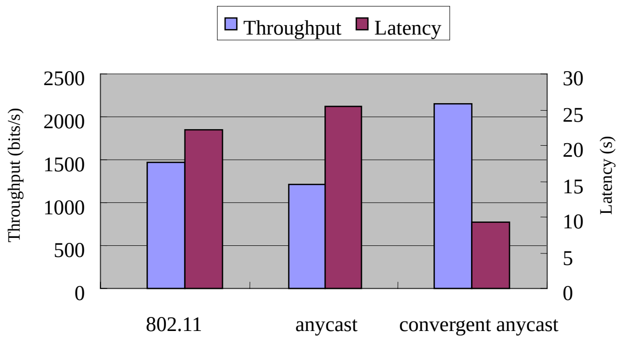

Figure 1: Throughput and latency simulation result in simple scenario.

3 Convergent MAC (CMAC)

In this section we present a high level overview of the protocol, a description of our assumptions, and a detailed description of the protocol.

3.1 Overview

All sensor nodes use unsynchronized random wakeup schedules with a pre-configured duty cycle, called the *idle duty cycle*. Upon observing an event, nodes use anycasting with greedy routing to forward packets towards the sink. While anycasting, neighbors closer to the destination respond before the neighbors nearer to the transmitter. Thus, anycasting prefers neighbors that are closer to the destination. The node that is selected as the receiver during anycasting, remains fully awake for a fixed duration. This ensures that anycast for subsequent packets will be received by this receiver or a neighbor that is even closer to the sink. Thus, the anycast progressively converges to unicast driven by the flow of data packets. This convergence typically completes within a few packet transmissions. Once the forwarding nodes are fixed for unicast, their wakeup schedules are matched to efficiently forward packets with low latency. After convergence, the wakeup schedule follows a different duty cycle called the *active duty cycle*. Thus CMAC avoids synchronization messages, minimizes the overhead of anycasting, and computes convergent routes and staggered wakeup schedules for low latency packet delivery while consuming much less energy.

3.2 Assumptions

In event triggered sensor networks, applications typically need to associate events with a location or a region in the sensor field. This requires all sensor nodes to know their own locations. As the network is static, such information can be obtained by using localization techniques right after the network deployment. In addition to its own location, we assume that each sensor knows the location of the sink and the location of its neighbors. Such information can be disseminated during localization. We also assume that the network has only one sink, but CMAC can be easily generalized for multiple sinks.

3.3 Detailed Protocol Description

In this section, we present details of the CMAC protocol. We discuss the anycasting technique, the mechanism for converging to unicasting, wakeup schedule synchronization on a single route, wakeup schedule synchronization on converging routes, and an extension of CMAC that deals with cross-route interference.

3.3.1 Anycasting Protocol

Due to the unsynchronized sleeping pattern, initially when an event is observed, unicast based forwarding may require a large number of attempts especially when the network is operating at a low duty cycle. To increase the chances of making progress in forwarding the packet closer to the sink, following GeRaF's approach, we use anycasting. This saves synchronization messages that are required in other protocols such as S-MAC [?, ?] and T-MAC [?]. Our implementation of anycasting uses a RTS-CTS exchange between the sender and the prospective receiver. 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 are closer to the sink. Following the principles of greedy forwarding [?], only nodes that are closer to the sink than the current sender are allowed to respond with a CTS. In order to decide when to send the CTS, the node coverage area that is closer to the sink than the node itself 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 2 shows an example where this area is divided into three regions. Nodes in region R_i send the CTS in time slot i. Nodes in the same region will choose a random time between 0 and k, to decide on a mini-slot to start transmitting the CTS, where k is the number of mini-slots per slot. If nodes overhear any traffic before transmitting CTS, they cancel the CTS 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 [?].

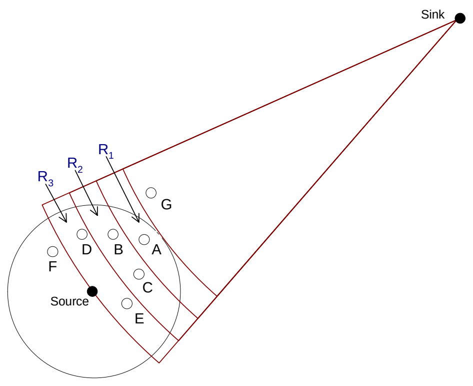

Figure 2: Coverage area closer than the sender is divided into regions according to the distance to the sink.

Although anycast obviates the need for synchronization messages, it has two 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, if there are no nodes in region R_1, the transmission will always be delayed because the CTS is delayed. Even when there are nodes in R_1, due to random backoff there could be a delay of more than one mini-slots before the CTS is transmitted.

3.3.2 Convergent Anycast

Anycast is a simple and robust scheme to quickly find a route to the sink. However, to avoid the inefficiency of anycasting and inefficiency due to higher number of hops, the route needs to converge to an optimal route on which unicast based traffic forwarding can be used to replace anycasting.

When the sender receives a CTS from a receiver, it first determines the region in which the receiver lies. If the receiver is in region R_1, the sender converges to unicasting to that receiver for subsequent packets. The two nodes then start following a staggered wakeup schedule with the active duty cycle. The details are described in Section 3.3.3. The sender indicates its willingness to synchronize to 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 to a synchronized staggered schedule.

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, the synchronization will timeout, and the nodes go back to the unsynchronized mode and start following the idle duty cycle.

The node that is selected as the receiver during anycasting, but is not selected to synchronize with the sender, remains fully awake for a fixed duration. This ensures that anycast for the next packet will be received by the current receiver or a neighbor that is even closer to the sink than the current receiver. If it cannot find a better next hop to synchronize with for a period, it will assume that there is no better node, and will start to synchronize with the latest receiver. Thus anycasting progressively converges to unicasting on a synchronized route.

3.3.3 Synchronization Schedule on a Single Route

The synchronization schedule is illustrated in Figure 3. 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. Figure 4 illustrates the synchronization schedules of some synchronized nodes. The receiving and transmitting slots are staggered from the source to the sink to minimize interference and reduce latency.

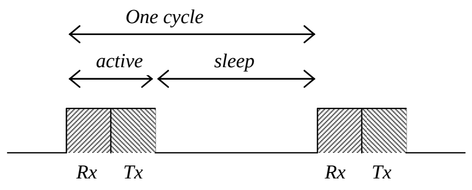

Figure 3: Synchronization Schedule.

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

1. *The sender is the source and is not synchronized.* When a sender is not synchronized, the schedule can be started at anytime. Figure 5 illustrates a scenario where an unsynchronized sender wants to

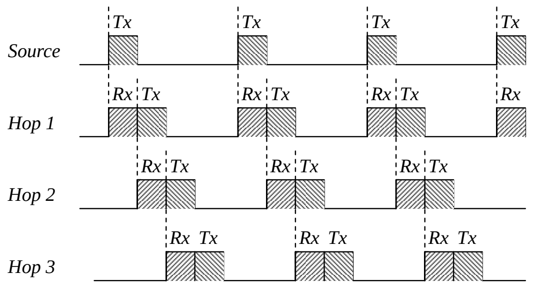

Figure 4: Staggered synchronization schedules.

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.

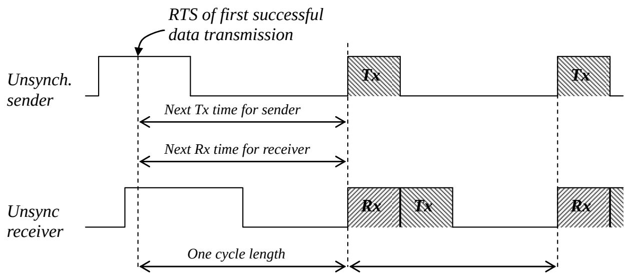

Figure 5: Unsynchronized sender synchronizes with the receiver. The schedule will be started as of the time the sender sends the RTS packet of the first successful data transmission.

2. *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 6). The receiver uses this offset to determine its staggered wakeup schedule to properly match the sender. Therefore even if the transmission failed for first few times, the receiver can still know the time to start the schedule.

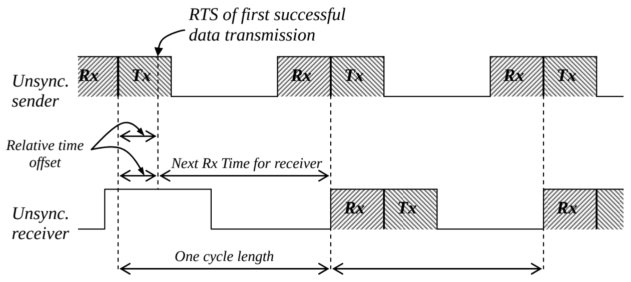

Figure 6: Synchronized sender synchronizes with the receiver. The sender will explicitly indicates the receiver the time elapse offset of its schedule in DATA header.

3.3.4 Synchronization Schedule on Multiple Converging Routes

As discussed before, our approach is to synchronize starting from the sources. This has the advantage of quick synchronization. However, a special case at nodes where routes merge, needs to be handled separately. Assume that a route from a source to the sink is already converged and synchronized. Another route merges to an intermediate node on the synchronized route. In this case, we have a scenario where a sender wants to synchronize with a receiver that is already synchronized to another sender. Such situations may arise in the case of simultaneous multiple event detections and in the case of mobile events. If the sender is not synchronized, the sender will follow 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 will know when to start its transmitting slot. If the sender is synchronized, there are several ways to deal with this scenario:

- The sender can match the receiver's schedule and ask its downstream 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 have a higher duty cycle than the active duty cycle (see Figure 7).
- The sender splits its receiving and transmitting slots to match with its upstream as well as downstream nodes as shown in Figure 8. We have implemented this approach for performance evaluation.

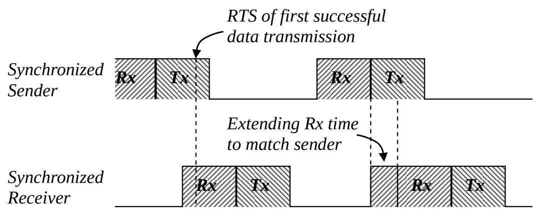
RTS of first successful data transmission
Synchronized Sender
Extending Rx time to match sender
Synchronized Receiver
Rx
Tx

Figure 7: The receiver extends its receiving slot to accommodate the new sender.

3.3.5 CMAC Extension

Although the staggering helps alleviate the contention between upstream nodes and downstream nodes, multiple merging routes can cause significant cross-route interference. Such interference is observed when multiple events occur simultaneously and also when the event is moving.

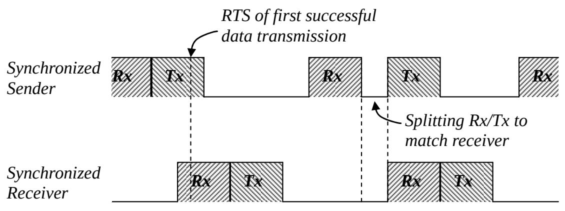
RTS of first successful data transmission
Synchronized Sender
Rx
Tx
Rx
Tx
Rx
Splitting Rx/Tx to match receiver
Synchronized Receiver
Rx
Tx
Rx
Tx

Figure 8: The sender splits its receiving and transmitting slots.

To address this problem, we propose a simple extension to CMAC. After converging from anycast to unicast, the nodes remain fully awake without any synchronization. So, nodes can transmit and receive at any time. Though this increases the energy consumption during active periods, only nodes forwarding data are required to remain fully awake. When the traffic ceases, the nodes go back to follow the unsynchronized idle duty cycle.

4 Performance Evaluation

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

- **Unicast:** Using RTS-CTS-DATA-ACK 802.11 protocol with optimal routing. Nodes have a duty cycle of 100%.
- **Anycast:** Using RTS-CTS-DATA-ACK anycasting protocol described in Section 3.3.1. Nodes are always active and they use anycast to forward packets.
- **GeRaF:** Just like Anycast, but nodes only work on 10% duty cycle. Our implementation has some differences from the description of GeRaF in [?]. First we do not use busy tone to detect if the channel is busy. Second, we use the anycast protocol described in 3.3.1, which is slightly different from the anycast protocol described in GeRaF [?].
- **CMAC:** Our proposed scheme described in Section 3. Nodes work on a 10% idle duty cycle. When node are synchronized, they will work on a 50% active duty cycle. The active time is split equally into receiving and transmitting slots. The length of the receiving and transmitting slot is just long enough to receive and transmit one packet.
- **CMAC-Ext:** Our proposed scheme with the extension described in Section 3.3.5. Nodes work on a 10% idle duty cycle. When nodes are synchronized, they remain active with a 100% duty cycle to receive and forward packets.

The radio transmission range is 250 meters, which is the default value in ns2. According to the Mica2 data sheet [?], 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 3.3.4, therefore the size of CTS for anycast is 14+6+2 = 22 bytes. The length of a time slot for receivers to reply the CTS should be long enough such that the probability of CTSs collision for nodes in the same region is low. We use 0.2ms in the following simulations. Table 4 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 1: Simulation parameters

We summarize the key results from our simulations below:

- CMAC outperforms all the other four protocols in terms of throughput, latency and energy saved in static event scenarios. With higher throughput and lower latency, CMAC can save up to 88.5% of energy spent in unicast.
- CMAC-Ext performs very close to unicast in terms of throughput and latency in static event networks, but can save up to 88% energy spent in unicast.
- In mobile event scenario, the performance of CMAC and CMAC-Ext is close to unicast under low traffic load. Under high traffic load, CMAC and CMAC-Ext achieve 80% and 87% throughput of unicast with similar latency, but can save about 84% of energy spent in unicast.
- Idle duty cycle has little impact on CMAC and CMAC-Ext.

4.1 Impact of Duty Cycles on CMAC

We evaluate the impact of the active duty cycle for CMAC protocol. CMAC uses a different duty cycle (active duty cycle) from the idle duty cycle after synchronizing. Figure 9 shows the throughput for different active duty cycle for a static event. It shows that at 50% duty cycle, CMAC has the best performance. With larger duty cycle, nodes have more time to receive and transmit packets. But when the duty cycle exceeds a threshold, nodes start to interfere with their upstream and downstream nodes as the interference range is typically larger than the transmission range (in the simulator, the interference range is twice the transmission range). A packet transmission can interfere with nodes 3 hops away because of the the CTS packet. Therefore it takes 3 more transmission time to forward a packet out of the interference range. Nodes working on 50% duty cycle, with 25% on receiving and 25% on transmitting, can avoid the interference. Therefore we use 50% active duty cycle in CMAC for the following simulation.

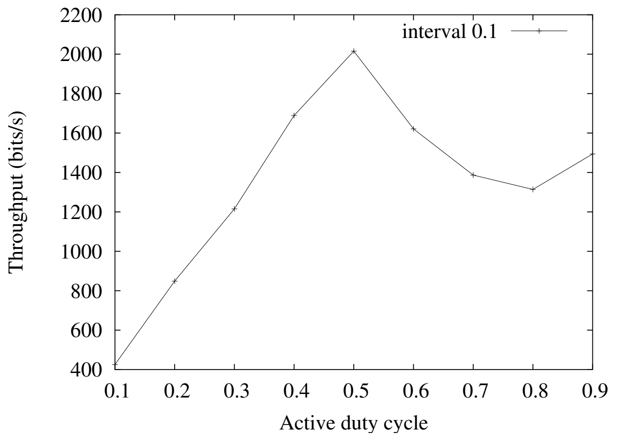

Figure 9: *The throughput of CMAC vs. active duty cycle.*At 50% active duty cycle, CMAC performs best by having longer time to transmit while reducing the interference between up/down stream nodes.

We study the impact of idle duty cycle on the CMAC protocol. We compare the performance of GeRaF, CMAC and CMAC-Ext using different idle duty cycles. Figure 10 shows the result in one static event network. It shows that the idle duty cycle has little impact on CMAC and CMAC-Ext. This is because after converging to unicast, nodes communicate only with their synchronized nodes, and only active duty cycle affects our protocols.

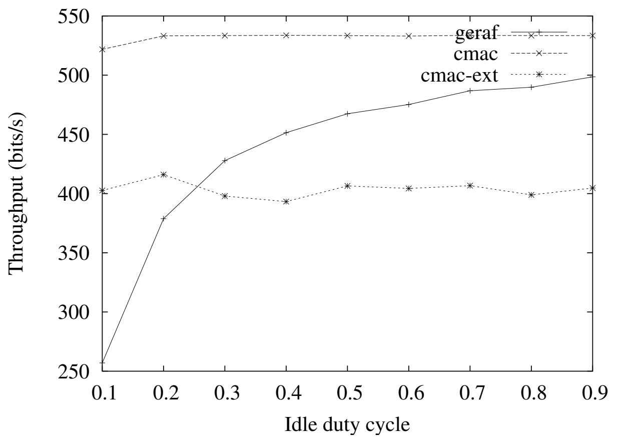

Figure 10: *The throughput of GeRaF, CMAC and CMAC-Ext vs. idle duty cycle.*Initial duty cycle has little impact on CMAC and CMAC-Ext since they will converge to unicast and use synchronized schedule to forward packets.

Figure 11 shows the convergence time of CMAC/CMAC-Ext using different idle duty cycles in networks with 100, 225, and 400 nodes in a one kilometer square area. Convergence time decreased when the idle duty cycle increased, or when the nodes density increased. Increasing idle duty cycle or the nodes density increases the number of active nodes at the same time. Therefore CMAC/CMAC-Ext has better chances to find a good node to synchronize with. We can see that at 10% idle duty cycle, CMAC typically converge after only a few seconds. Packets are still being forwarded to sink while converging, therefore the convergence time has little impact on the performance of our protocols. Figure 10 confirms the performance of our protocols are not affect by the idle duty cycle.

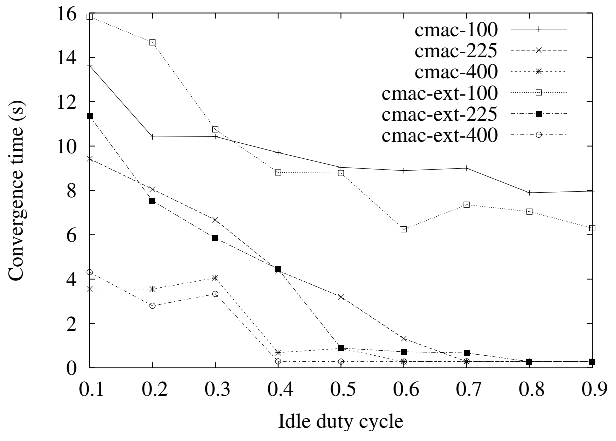

Figure 11: *The converging time of CMAC vs. idle duty cycle in networks with different node density.*The higher the idle duty cycle or node density, the lower the convergence time.

4.2 Static Event

We use a grid network to evaluate the five protocols. The grid size is 20 nodes by 20 nodes, and the distance between two nodes is 50 meters. First we evaluate these protocols in one static event network. The source that detect the event is located at the left-bottom corner, and the sink is located at the right-upper corner. We vary the traffic load from 2 packets per second to 10 packet per second. When the data rate is lower than 5 packets per second, all protocols can deliver 100% of the packets with low latency except GeRaF, therefore we only show the simulation results for data rate between 5 packets per second and 10 packets per second. Each result is the average of 5 independent simulations running for 400 seconds each. Figure 12, 13 and 14 show the simulation result of throughput, latency and energy consumption of different mechanisms.

In low traffic load networks, the throughput and latency of these protocols are almost the same except GeRaF, but the energy consumption in our protocols

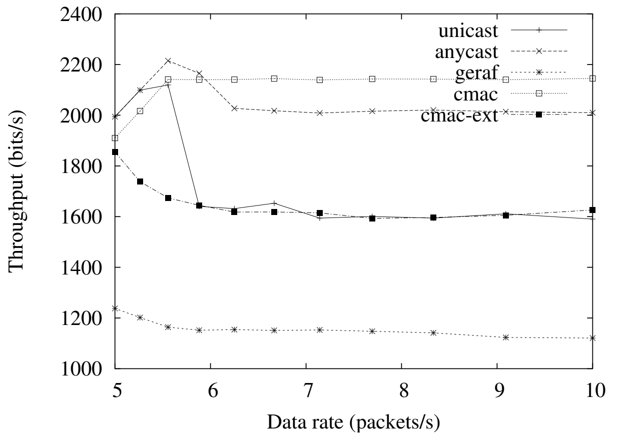

Figure 12: Throughput of protocols with one static event. CMAC outperforms all other four protocols because of staggered receiving and transmitting slots, and CMAC-Ext performs close to unicast.

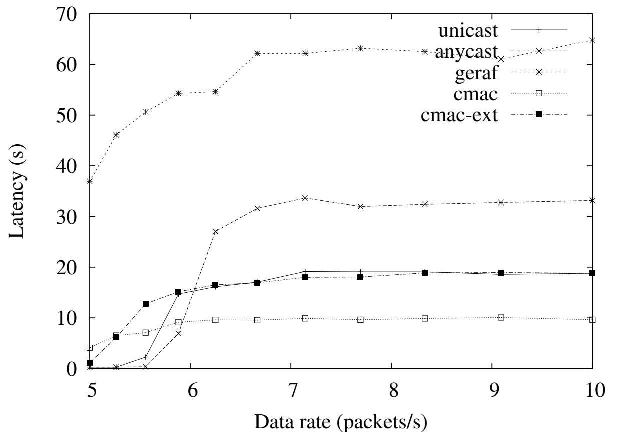

Figure 13: End-to-end latency of protocols with one static event. CMAC has the lowest end-to-end delay and CMAC-Ext performs close to unicast.

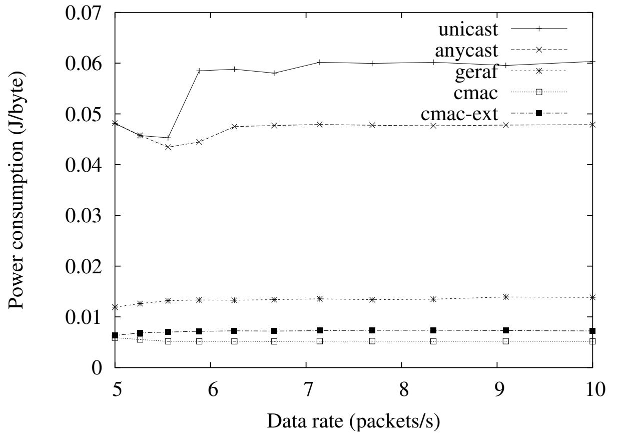

Figure 14: Normalized energy consumption. The amount of energy spent per byte.

are the lowest. GeRaF suffers from low throughput and high latency because the forwarding path is not fixed. Without high node density, the packets may get dropped in the middle of a route because all its neighboring nodes may be sleeping. When the traffic load is high, CMAC outperforms all other protocols in every aspect. This is because CMAC staggers the receiving and transmitting slots, thus, reducing the contention between upstream and downstream nodes. With higher throughput and lower latency, CMAC can save up to 88.5% of energy spent in unicast while working on 10% idle duty cycle.

We can see from the simulation results that the throughput of anycast protocol is better than the unicast protocol, but with higher delay. This is because in a grid network, nodes have multiple choices for next hop to forward packets to. As long as one receiver is available to receive the packet, the packet can be forwarded, thus increase the utilization of the channel. Since it does not use optimal route, the average hop count will be high, too. Therefore the delay in anycast is higher than unicast. Figure 15 shows the average hop count in different protocols. The hop count of the optimal route is 6, which is the hop count used in unicast. CMAC and CMAC-Ext depend on the anycast protocol to converge to a unicast path when the next hop is located in the farthest region, therefore they have lower hop count compared to the GeRaF.

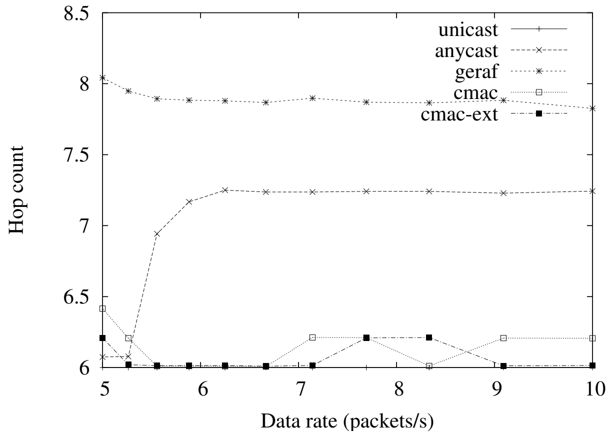

Figure 15: Average hop count of packet forwarding path in static event network.

4.3 Mobile Event

We evaluate the performance of these five protocols in mobile event scenario in the grid network. An event moves randomly in the network for 293 seconds at the speed of 10 m/s. The sensing range of sensors is configured to 35.5 meters. This ensures that the event is always in range of at least one sensor. Figure 16, 17 and 18 show the simulation results of throughput, latency and energy consumption of different protocols. Figure 19 shows the average hop count for different protocols.

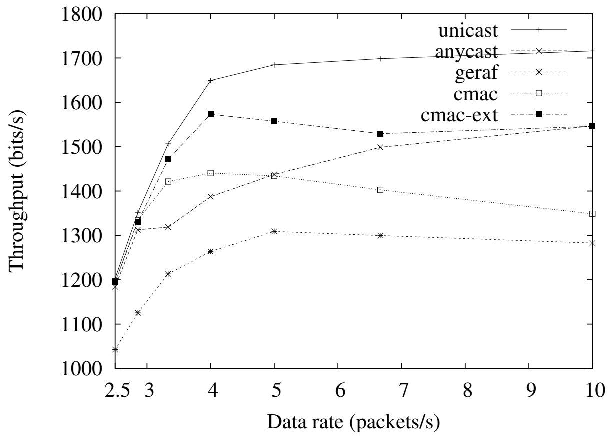

Figure 16: *Average throughput of protocols with one mobile event.*CMAC and CMAC-Ext perform close to unicast in lower traffic load network, and can achieve 80% to 87% throughput in high traffic load and high intereference network.

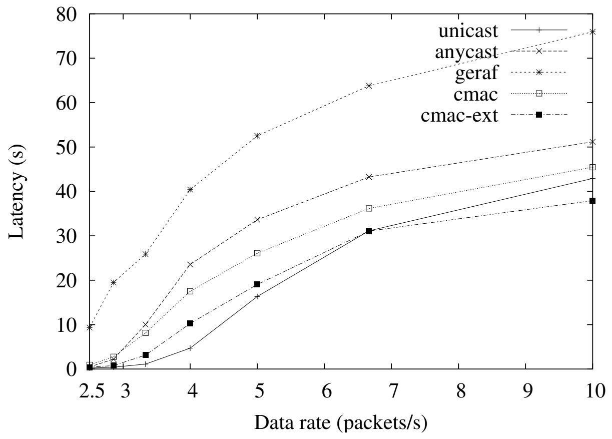

Figure 17: *Average end-to-end latency delay with one mobile event.*The latency of CMAC-Ext is very close to unicast.

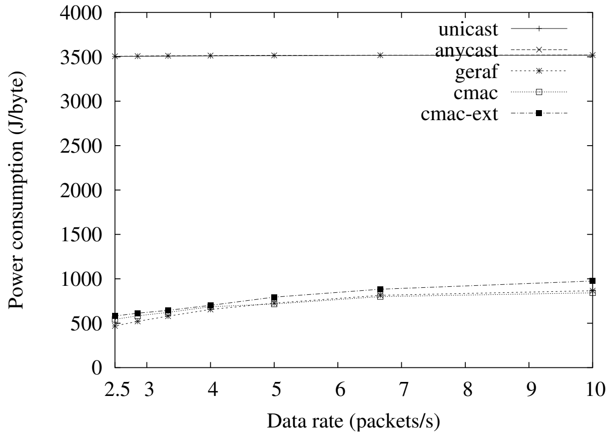

Figure 18: Normalized energy consumption, the amount of energy spent per byte.

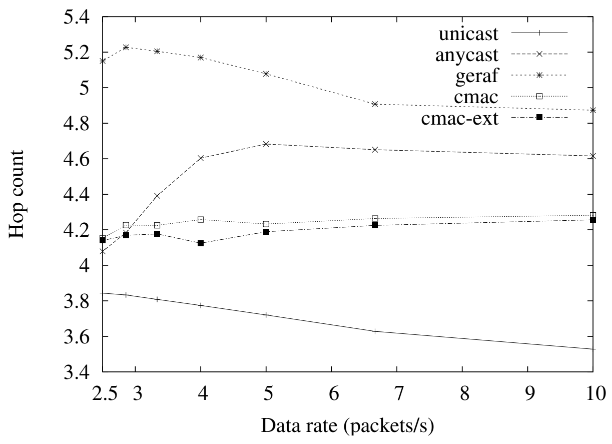

Figure 19: Average hop count of packet forwarding path in mobile event network.

In mobile event networks, CMAC and CMAC-Ext still use the least energy, while achieving 80% to 87% of throughput in unicast with a slighly higher latency. The reason for the performance drop of CMAC and CMAC-Ext is because the event may move out of a sensor's sensing range before the route converges. Therefore the benefit of converging to unicast route is reduced. However, unicast achieves highest throughput with very high power consumption as nodes remain awake with a 100% duty cycle. When the network is idle, unicast will spend 10 times more energy than CMAC/CMAC-Ext. When there are events, unicast provides only 15% higher throughput than CMAC-Ext, but spends 4 to 6 times more energy than CMAC/CMAC-Ext. Therefore CMAC/CMAC-Ext are still the best for networks that require longer network lifetime.

We can also see that CMAC-Ext performs better than CMAC for mobile event. This is because of high interference from neighboring data flows. In the mobile event network, there may be 4 or more neighboring nodes transmitting packets concurrently, which causes very high contention and interference. As described in Section 3.3.5, the staggered receiving and transmitting slots may reduce the channel utilization in high contention scenario. CMAC-Ext improves the performance of throughput and latency by allowing more nodes to contend for the channel at the same time.

In summary, CMAC and CMAC-Ext perform very well in both static and mobile event scenarios, and in both high and low traffic load networks. CMAC outperforms unicast and anycast and still can save about 88% energy in static event networks. In high traffic load with moving event scenario, CMAC-Ext performs slightly better than anycast, and achieves 90% throughout of unicast in high traffic load network, while saving 73% energy of unicast and anycast. Therefore, CMAC/CMAC-Ext are very suitable for extending the network lifetime of sensor networks.

5 Related Work

Sensor networks can increase the network lifetime 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 the packets can be forwarded to destination despite 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 [?], nodes exchange their wakeup and sleep schedules with their neighbors before starting their periodic listening and sleeping schedule. This can significantly reduce the time spent on idle listening and therefore save energy. In later work [?], the authors extend S-MAC abilities to use adaptive listening to reduce the latency. Nodes overhearing their neighbor's RTS or CTS packets will wake up for a short duration at the end of the transmission to check if there is data for them.

T-MAC [?] uses the same mechanism as S-MAC to synchronize nodes, but save more energy by using adaptive duty cycle 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 idle listening the channel for entire listening time, nodes will go back to sleep if there is no event happened.

DMAC [?] and [?] 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 message .

In [?], nodes in power-save mode are sleeping most of the time, and wake up periodically to check for pending data. Nodes are triggered by communication events, such as route discovery messages or data packets, to switch from power-save mode to active mode. In active mode, nodes are active with 100% duty cycle to receive and transmit packets. Nodes will go back to power-save mode if there is no traffic for a certain duration.

GAF [?] saves energy by putting nodes that are redundant in the aspect of routing into sleep. In GAF, nodes are divided into virtual grids. Every node in a grid is in the radio range of nodes in adjacent grids. Therefore, one active node in each grid is enough to maintain the network connectivity. Nodes in GAF can be in three states: sleeping, discovery and active. All nodes are in discovery state initially, and set a timer to broadcast a discovery message. If nodes receive a discovery message from other nodes before they broadcast their discovery message, they will go to sleeping state; otherwise they will enter the active state. In active state, nodes will set a timer to define how long they will stay active, and periodically broadcast the discovery message. When the timer expires, the active node will go back to discovery state.

In B-MAC [?], 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.

PEAS [?] provides a resilient, long-lived sensor network for a unreliable environment. Nodes in PEAS go to sleep and wakeup periodically. When a node wakes up, it sends a PROBE message to probe if any of its neighbors are awake. Nodes that receive the PROBE message will broadcast a REPLY message to notify that they are awake. Since some of the nodes in coverage area are awake, it's not necessary for that node to be active, so the node will go back to sleep for a period according to the information gathered from the REPLY message.

GeRaF [?][?] eliminates the synchronization and probing message required by protocols mentioned before by using anycast. In GeRaF, nodes know the location of their neighbors and the sink. When a node wants to send packets to the sink, it broadcasts an RTS to all its neighbors. Nodes that receive the RTS will reply with a CTS packet according to their distance to the sink. Nodes that are closer to the sink will send the CTS first. When the source node receives the CTS, it sends the DATA packet to the node from which the CTS was sent. Other nodes overhearing the data packet cancel their CTS transmission. This mechanism is simple because nodes don't 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.

Anycasting: GeRaF uses anycast to forward data. Anycast allows a node to transmit packets to any node among a subset of nodes. When a node transmits a packet, as long as one of its neighbors acknowledges it, the packet can be forwarded. [?] proposed another anycast protocol using similar handshaking protocol as in 802.11 DCF. The sender broadcasts an MRTS packet containing multiple next hops provided by the routing protocol. These next hops are assigned with a priority order according to some metric, such as their address position in MRTS packet, or their hop count to the destination. The receivers will reply with a CTS according to their priority. The node with the highest priority sends the CTS after an SIFS time. The second node sends the CTS after a period equal to the time to transmit a CTS plus 3 SIFS time to ensure that CTSs do not collide. When the sender receives the CTS, it transmits the DATA packet to the sender of the CTS. Other nodes overhearing the DATA packet will cancel their CTS transmission.

6 Conclusion

This paper proposes a MAC layer approach for conserving energy in sensor networks. Based on the current protocols for energy conservation, we make two key observations to motivate our design. First, synchronized wakeup is wasteful in sensor networks due to the overhead of synchronization messages. Second, anycasting has high latency in non-synchronized sensor networks. Based on these observations, we design the CMAC protocol. During idle periods, the nodes are unsynchronized and use a small duty cycle (termed idle duty cycle) to randomly wakeup and sleep. CMAC initially uses anycasting due to lack of synchronization across nodes, but quickly converges to unicasting to reduce latency. The converged routes switch to using an active duty cycle with staggered synchronized schedules. We found that a 50% active duty cycle results in the best throughput. 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 11.5% energy consumption of unicast. For mobile events, CMAC-Ext achieves 90% to 99% throughput of unicast with similar latency while spends only 17% to 28% energy of unicast and anycast. Based on our study, we conclude that CMAC is highly suited for event triggered sensor networks that require long network lifetime. Currently we are planning 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, 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] Sental 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] A. El-Hoiydi and J.D. Decotignie, " WiseMAC: An Ultra Low Power MAC Protocol for Multi-Hop Wireless Sensor Networks," in *Algosensors*, July 2004.

[10] Wei Ye, John Heidemann, and Deborah Estrin, " An energy-efficient MAC protocol for wireless sensor networks," 2002.

[11] Wei Ye, John Heidemann, and Deborah Estrin, " Medium access control with coordinated adaptive sleeping for wireless sensor networks," 2004.

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

[13] Michele Zorzi and Ramesh 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.

[14] Michele Zorzi and Ramesh 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.

[15] Joseph Polastre, Jason Hill, and David 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.

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

[17] Ya Xu, John S. Heidemann, and Deborah Estrin, "Geography-informed energy conservation for Ad Hoc routing," in *Mobile Computing and Networking*, 2001, pp. 70–84.

[18] 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.

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

[20] Gang Lu, Bhaskar Krishnamachari, and Cauligi 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.

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

[22] " Crossbow Technology," http://www.xbow.com/.

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

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

[25] 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.