# WaveScheduling: Energy-Efficient Data Dissemination for Sensor Networks

Niki Trigoni, Yong Yao, Alan Demers, Johannes Gehrke Cornell University Ithaca, New York 14850 {niki,yao,ademers,johannes}.cs.cornell.edu Rajmohan Rajaraman Northeastern University Boston, Massachusetts 02115 rraj@ccs.neu.edu

## Abstract

Sensor networks are being increasingly deployed for diverse monitoring applications. Event data are collected at various sensors and sent to selected storage nodes for further in-network processing. Since sensor nodes have strong constraints on their energy usage, this data transfer needs to be energy-efficient to maximize network lifetime. In this paper, we propose a novel methodology for trading energy versus latency in sensor database systems. We propose a new protocol that carefully schedules message transmissions so as to avoid collisions at the MAC layer. Since all nodes adhere to the schedule, their radios can be off most of the time and they only wake up during welldefined time intervals. We show how routing protocols can be optimized to interact symbiotically with the scheduling decisions, resulting in significant energy savings at the cost of higher latency. We demonstrate the effectiveness of our approach by means of a thorough simulation study.

## 1 Introduction

Sensor networks consisting of small nodes with sensing, computation and communication capabilities are becoming ubiquitous. A powerful paradigm that has emerged recently views a sensor network as a distributed Sensor-DBMS and allows users to extract information by injecting declarative queries in a variant of SQL. In deploying a SensorDBMS one should consider important limitations of sensor nodes on computation, communication and power consumption. Energy is the most valuable resource for unattended battery-powered nodes. Since radio communication consumes most of the available power, SensorDBMSs need energy-efficient data-dissemination techniques in order to extend their lifetime.

An important communication pattern within sensor networks is the sending of sensor readings to a designated

Copyright 2004, held by the author(s)

Proceedings of the First Workshop on Data Management for Sensor Networks (DMSN 2004), Toronto, Canada, August 30th, 2004. http://db.cs.pitt.edu/dmsn04/ sensor node. Let us give two examples where this pattern arises. First, consider a heterogeneous sensor network with two types of sensor nodes: many small-scale source nodes with low-power multi-hop communication capabilities, and a few powerful gateway nodes connected to the Internet. In this setup, data flows from the sources to the gateway nodes. Our second example is motivated by resource savings through in-network processing. In-network processing algorithms coordinate data collection and processing in the network at designated nodes called view nodes [1, 2]. Data flows from sources to relevant view nodes for further processing. For example, in a sensor network that monitors a remote island and records the movements of different types of animals, each view node could be responsible for storing the detection records (and computing tracks) for a given type of animal.

Since power is a major resource constraint, we would like this data flow between sources and view nodes to be as power-efficient as possible; in particular, for non-timecritical applications, we would like to trade message latency versus power usage as events are routed from the sensor nodes where they originated to the respective view nodes.

In order to achieve energy-efficient data flows between sources and view nodes, we address several challenges intrinsic to ad hoc network communication: minimizing collisions at the MAC layer, managing radios in a powerefficient manner, and selecting energy-efficient routes. In this paper we consider data dissemination strategies that avoid collisions (and message retransmissions) at the cost of higher message latency. We carefully coordinate transmissions between nodes, allowing them to turn their radios off most of the time. Current generation radios consume nearly as much power when listening or receiving as when transmitting (typical idle:receive:transmit ratios are 1:1.2:1.7 [3], 1:2:2.5 [4], and 1:1.05:1.4 [5]). Thus, the ability to turn them off when not needed yields significant energy savings.

The remainder of this paper is organized as follows. Section 2 enumerates several variants of scheduling problems and discusses their complexity. Section 3 presents our scheduling algorithm and highlights its close interaction with the routing layer. A thorough experimental evaluation of the proposed algorithm and competing approaches is presented in Section 4. We discuss related work in Section 5 and draw our conclusions in Section 6.

# 2 Problem space

With coordinated scheduling, a data dissemination protocol in a sensor network has two components: a scheduling algorithm that activates network edges so that their transmissions do not interfere with one another, and a routing algorithm that selects routes for individual messages. Two important performance metrics are energy consumption and message latency. In this section, we consider each of these metrics and sketch complexity results for the following optimization problems: (i) finding an optimal pair of routing and scheduling algorithms; (ii) finding an optimal routing algorithm for a given schedule; (iii) finding an optimal schedule for a given collection of routes. Full proofs of these results can be found in an extended version of the paper [6].

The underlying framework for our optimization problems is as follows. We assume the sensor nodes form a multi-hop wireless network embedded in the plane. For simplicity, we assume the node radios have identical ranges of one unit. Thus, the nodes form a *unit disk graph*: two nodes are connected by an edge iff the Euclidean distance between them is at most 1. We represent the communication workload by the rate of message generation at each node *i*, given by  $r_i$ , together with a probability distribution  $p_{ij}$ , giving the probability that a message generated at node *i* is destined for node *j*.

**Energy minimization.** In the energy minimization problem, we are given a communication workload among the sensor nodes and view servers, and our goal is to determine a data dissemination scheme that minimizes the energy consumed in delivering all messages within a bounded delay. In our model, we assume that the energy consumed when a network edge is activated is  $(\alpha + \beta m)$ , where  $\alpha$  is a fixed start-up cost for turning the radio on,  $\beta$  is the per-message transmission and reception cost, and m is the number of messages sent during the activation.

**Theorem 2.1** For any  $\alpha > 0$  and  $\beta > 0$ , finding an optimal routing-scheduling pair to minimize energy is NP-hard, even when there is only one view server. It is also NP-hard to determine an energy-optimal activation schedule given a fixed set of routes. The problem of finding a set of energyoptimal routes given an activation schedule can be solved in polynomial time.

Latency minimization. In the *latency minimization* problem, we are given a communication workload and seek a data dissemination protocol that minimizes average message propagation latency. It is already known that minimizing latency in an *ad-hoc* wireless network is NP-hard even for the special case where nodes exchange messages only with their neighbors [7]. This reduction can be extended to unit disk graphs.

**Theorem 2.2** Finding a routing-scheduling pair that minimizes latency is NP-hard. It is also NP-hard to determine an optimal activation schedule given a fixed set of routes. A set of latency-optimal routes for a given activation schedule can be obtained in polynomial time.

These results indicate that the general problem of designing an optimal data dissemination protocol, given an arbitrary sensor workload, is intractable. In this paper, we focus on one element of the design space, namely that of first developing an interference-free schedule for edge activation, and then designing delay- or energy- optimal routes given this schedule.

# 3 Wave Scheduling and Routing

In this section, we focus on developing a schedule for edge activations, and then designing optimal routes given this schedule. Our scheduling mechanism is defined over a simple partitioning of the network, which we first describe in Section 3.1. We then select a class of periodic schedules, presented in Section 3.2, which are aimed at avoiding collisions at the MAC layer. Finally, in Section 3.3, we present energy-based and delay-based routing protocols that optimize the relevant metric for a given schedule.

## 3.1 Partitioning

Our scheduling mechanism is layered on top of a protocol like GAF [8], which partitions nodes into cells and periodically elects a single leader node for each nonempty cell. Nodes determine the cell that they belong to by using distributed localization techniques [9, 10]; experiments have shown that GAF is robust to somewhat inaccurate position information [8]. The size of each cell is set so that a node anywhere in a cell can communicate directly with nodes in any of its four horizontal and vertical neighbor cells. This constrains the side of a cell to have length L at most  $R/\sqrt{5}$ . where R is the transmission range of a node. Since only leaders are engaged in inter-cell message routing, the remaining nodes may turn off their radios most of the time, achieving significant energy savings. The schedules that we will propose in this section exploit the GAF topology control scheme in order to achieve further energy savings. They leverage the abstraction of partitioning irregularly positioned nodes into cells organized in a rectilinear grid and focus on coordinating inter-cell communication. In what follows, we will refer to cells as *supernodes* or simply nodes.

For convenience of exposition, we assume here that the rectilinear grid is a square. Let N denote the number of supernodes along an edge of the grid. We identify the supernodes by their coordinates (i, j); for example (0, 0) refers to the node at the southwest corner of the network. Thus, (i + 1, j), (i, j + 1), (i - 1, j), and (i, j - 1) are the east, north, west, and south neighbors, respectively, of node (i, j), for  $i, j \in [0, N)$ .

## 3.2 Wave Scheduling: Algorithms

Given a set of supernodes arranged in a rectilinear grid, we propose a class of periodic activation schedules that conserves energy by (i) avoiding interference at the MAC layer and (ii) allowing supernodes to turn off their radios whenever they are not sending or receiving messages. In these schedules, which we call wave schedules, every (directed) edge of the rectilinear grid is activated periodically at well-defined communication intervals, called sendreceive intervals. For any two neighboring supernodes A and B, the edge  $A \rightarrow B$  is activated in the send-receive intervals  $[t + iP, t + iP + \delta]$ , for every  $i \ge 0$ , where t is the first time the edge is activated and P is the period of the schedule. We now elaborate on the edge activation



Figure 1: SimpleWave

step and then present two wave schedules: SimpleWave and PipelinedWave.

**Edge activation**. An edge activation  $A \rightarrow B$  consists of a contention-based and a collision-free period. During the contention-based period, all nodes within the cell A turn on their radios in order to run the GAF protocol (GAF only runs locally in cell A). They check whether the leader has enough energy reserves to continue assuming the leadership role. If the leader is energy-drained, a re-election protocol selects the new leader. Messages in the queue of the old leader, as well as inter-cell routing information, are transfered to the new leader. The remaining nodes then send their sensor readings, which were generated since the previous GAF period, to the leader of the cell. Contention resolution MAC protocols work very well in avoiding intracell contention, since all nodes in the cell are within communication range and there are no occurences of the hidden terminal problem. This adapted version of the GAF protocol is more energy-efficient than the original GAF scheme, because it avoids interference caused by concurrent leader reelection in consecutive cells.

The collision-free period of an edge activation  $A \rightarrow B$ is used in order to route messages from the leader of Ato the leader of B. During that period both leaders of A and B (referred to simply as A and B) turn on their radios preparing for message transmission and reception respectively. If A has no data messages to send, it sends a special Nothing ToSend (NTS) message to node B, which allows both nodes to turn off their radios without having to wait until the end of the send-receive interval. As we will show in the experimental section, the use of NTS messages offers significant energy savings since it adjusts the node duty cycle to its local traffic. Since in the collision-free periods there is no interference at the wireless medium, it is not necessary to exchange RTS and CTS messages prior to sending a regular data message (or an NTS message). A data (or NTS) message is simply followed by an ACK. The first data or NTS message that A sends to B (and its ACK) can be used in order to resynchronize the clocks of the two nodes for the next activation of edge  $A \rightarrow B$ . If the synchronization error between two neighbor nodes at the beginning of the collision-free period is bound by emsecs, we set the receiver B to wake up e msecs earlier

than scheduled according to its local clock. Synchronization issues are discussed in more detail in the end of this section.

In the remainder of the paper, by edge activation we mainly refer to the collision-free period of the edge activation used for inter-cell communication. The ratio of the collision-free period to the contention-based period depends on the traffic patterns of the application. For instance, for traffic workloads with messages following multiple hops before reaching the destination, the collisionfree (inter-cell communication) period should dominate the contention-based (GAF) period.

**SimpleWave**. The intuition behind wave schedules is to coordinate message propagation in north, east, south and west phases. For instance, during the east phase, only edges of the form  $(i, j) \rightarrow (i + 1, j)$  are activated sending messages along the east direction. Owing to interference, however, we cannot schedule all of the edges along the east direction. If  $\Delta$  denotes the ratio of the interference range to the transmission range, then a sufficient condition for transmissions from two supernodes (i, j) and  $(i_1, j_1)$  to avoid interference is the following:

$$\sqrt{(i-i_1-1)^2+(j-j_1-1)^2}\cdot L\geq \Delta\cdot R$$

In particular, if we consider two supernodes (i, j) and  $(i_1, j)$ , then their transmissions do not interfere it  $i - i_1 - 1 \ge \Delta R/L$ . Since  $i - i_1 - 1$  is an integer, we obtain that the two supernodes can transmit simultaneously if  $i - i_1 \ge \left\lceil \Delta \cdot R/L \right\rceil + 1$ , which we denote by g. If we adopt the IEEE 802.11 settings of R = 250m and  $\Delta = 550/250$ , and set L to its minimum value  $R/\sqrt{5}$ , we obtain that g = 6.

In the SimpleWave schedule, we schedule together edges that are q positions apart. Figure 1 illustrates the Simple-Wave schedule on a  $10 \times 10$  network, with R = 250m,  $\Delta = 550/250$ , setting L to a round number of 100m (instead of its minimum value  $R/\sqrt{5}$ , yielding q = 7. The north phase starts at time 1 and it lasts for 51 send-receive intervals during which every north edge is activated exactly once. The following east phase starts at time 52. Notice that only two nodes of the first column ((0,0) and (0,7))are sending concurrently to the east, which are spaced apart by 7 hops. In the next interval (time 53) the pattern shifts east by one cell. Only when the wave has propagated to the eighth column (time 59) does it no longer interfere with node communication in the first two columns. Note that at time 59 it becomes possible to schedule concurrently four edges:  $(7,0) \rightarrow (8,0), (7,7) \rightarrow (8,7), (0,1) \rightarrow (1,1)$ and  $(0, 8) \to (1, 8)$ .

There are variants of the SimpleWave algorithm defined above, differing by the order in which wave directions are scheduled. We refer to these as the (N, E, S, W), (N, W, S, E), (N, S, E, W), and so forth. The variants are logically equivalent, but the choice of scheduling variant affects the choice of routes, as will be explained in detail in section 3.3. The period of a SimpleWave depends on the size of the network. Each phase takes  $(N-1) + (g-1) \cdot g$ send-receive intervals and the entire wave period lasts for  $4 * ((N-1) + (g-1) \cdot g)$  intervals. This is not a desirable property, because it prevents the distributed deployment of the algorithm in a dynamic network. When a new supernode (cell) joins (or leaves) the network, it affects the



Figure 2: PipelinedWave

wave period and therefore the activation times of all the other supernodes. In addition, in order to identify the activation time of its adjacent edges a supernode should know its location within the network, as well as the size of the network. Another important downside of the *SimpleWave* algorithm is that it underutilizes the capacity of the network. For instance notice in Figure 1 that at time 1, it activates only two north going edges, whereas one could identify two additional edges that could be activated concurrently without causing any interference.

**PipelinedWave**. This algorithm is motivated by the need for schedules that can be deployed in a distributed and scalable manner, and that make a good use of the network capacity. Conceptually, a network can be divided in a number of fixed-size  $(g \times g)$  squares of  $g^2$  supernodes each, where all squares have identical schedules. In such a network, the schedule of the incident edges of a node is determined by its relative location in the square. Since all edges within the same square interfere with one another, we can only schedule one edge at a time. In effect, we partition all the edges of the network into a collection of maximal independent sets, each independent set corresponding to a set of edges that can be simultaneously activated without interference. The period of the resultant schedule is  $4g^2$  send-receive intervals. This means that for pipelined waves, new nodes can join the network and schedule themselves without affecting the schedules of existing nodes. If a supernode joins an existing square, it waits for at most one period in order to interact with its neighbors and locally determine its location with respect to them and therefore its local coordinates within the square. By overhearing the schedules of its immediate neighbors it determines the time at which it should schedule itself in each direction. A similar local interaction occurs when a new supernode joins the network initializing a new square. When a node leaves the network, the schedules of the remaining supernodes do not change.

Note that in the *PipelinedWave* algorithm two edges are scheduled concurrently if they have the same direction and the sender nodes (and the receiver nodes) have exactly the same local coordinates within a  $g \times g$  square. This implies that the algorithm avoids all interference at the MAC layer. It schedules a maximum number of non-interfering edges at each send-receive interval thus increasing the network capacity with respect to the *SimpleWave* algorithm. It is easily deployable in a distributed manner, since local coordination suffices for scheduling a new supernode. Finally, it is scalable because the node schedules are not affected by the size of the network.

A modified version of the *PipelinedWave* algorithm does not define identical schedules for each square, but schedules shifted by q positions with respect to the schedules of the four neighbor squares. More specifically, the east wave of a square is shifted q positions (send-receive intervals) earlier than the east wave of the west neighbor square, the north wave is shifted q positions earlier than the north wave of the south neighbor square etc. A snapshot of the modified PipelinedWave algorithm (during the east phase) is shown in Figure 2. The east phase in a given (dotted) square proceeds by shifting one edge to the right and moving to the row below when the entire row of the square is traversed. Notice that by the time an entire row is traversed in a given square, the respective row of the right neighbor square just starts being traversed. The new pipelined algorithm decreases the latency of message delivery at the square boundaries; this will become evident when we describe *delay-based* routing in Section 3.3. The south, west and north phases are scheduled in a similar manner. This improved *PipelinedWave* is the schedule evaluated in our experiments in Section 4.

Another tunable parameter in *PipelinedWave* is the number of send-receive intervals for each direction (phase) before the wave switches to another direction. Our experiments show that this parameter, referred to as *step*, has no noticeable impact on the performance of the wave schedule [6]. In Section 4, we evaluate the variant of *PipelinedWave* with step=1.

Synchronization. We briefly discuss two synchronization requirements imposed by wave schedules: i) neighbor nodes must have the same notion of time regarding their communication slot and ii) nodes in the *close* neighborhood must be well synchronized so that only edges at least q positions away are scheduled simultaneously. Acknowledging that perfect time synchronization is hard to achieve, we relax the initial requirements and propose a fault-tolerant version of wave schedules. If the drift between two neighbor clocks does not exceed  $\epsilon$ , nodes that are q positions away from each other are synchronized within  $q\epsilon$ . In every edge activation, we schedule the receiver to turn on the radio  $\epsilon$ time units earlier than the scheduled time according to its local clock. In order to ensure that there is going to be no interference due to the clock errors, we can increase the distance between two non-interfering edge activations (e.g. from 7 to 8). Notice that although a perfectly aligned wave schedule implies global synchronization, a reasonable implementation of waves is achievable by ensuring that nodes are well-synchronized with neighbors within interference range.

Recently proposed synchronization protocols for sensor networks (e.g., RBS [11] and TPSN [12]) provide tight synchronization bounds (e.g., 0.02ms for neighbor nodes [12]) and exhibit good multi-hop behavior. Their performance however is bound to decay for very large networks (an open problem that we discuss in Section 4); in this case we assume that a few GPS-equipped nodes will undertake the synchronization task for their local regions.

## 3.3 Routing

The proposed wave schedules are TDMA-based MAC protocols that assign periodic transmission slots to intercell communication. Wave schedules are general-purpose energy-efficient MAC protocols that can potentially be combined with arbitrary routing protocols. In this section we consider two important metrics for evaluating the efficiency of a routing algorithm, namely *node energy consumption* and *message propagation latency*. Note that energy-optimal routes do not depend on the underlying wave schedule, whereas latency-optimal routes are intrinsically coupled with it.

**Energy-based routing**. As noted in Section 2, minimum energy routing is achieved by routing along shortest hop paths. We adopt a simple flooding approach that evaluates minimum-hop paths from all nodes in the network to a given view node. Flooding initiated at a view node results in the construction of a tree connecting all supernodes to the root (view) node, as described in [13]. Since we consider more than one view, the minimum-hop routes form a forest of trees built on top of the grid overlay.

Each node maintains an in-memory routing table of size proportional to the number of view servers. For each view server, the routing table includes a 2-bit entry giving the direction of the next hop towards the view. This simple approach works even in the presence of "holes" (empty cells), as is shown by Madden et al. [13]. Dynamic node failures (which manifest themselves as the appearance of new holes) can be dealt with by a local flooding phase to repair affected routes, as in AODV, or by introduction of a greedy face-routing mode as in GPSR [14, 15]. Alternatively, a node that fails to deliver a message may store it in memory until the next flooding phase that reconstructs the tree.

Delay-based routing. We propose a *delay-based* routing algorithm that, given a certain wave schedule, minimizes message latency between a pair of source and view nodes. Each node C maintains a routing table, that contains for each view V and each neighbor N a triple  $\langle V, N, d \rangle$ , where d is the latency of the minimum-latency path from C to V among all paths with the next-hop being N that C is presently aware of. On updating a routing entry, node Calso sends the information  $\langle V, N, d \rangle$  to its neighbors. On the receipt of such a message, neighbor  $N^*$  of C does the following: i) it evaluates the time dt that a message sent over  $N^* \to C$  remains at C before being forwarded with the next wave via  $C \to N$  towards view V; ii) if an entry  $\langle V, C, d' \rangle$  with d' < d + dt exists in the routing table of  $N^*$ , then the routing message is dropped - otherwise, the routing entry is replaced by  $\langle V, C, d + dt \rangle$ .

When the above distributed algorithm converges, every node has determined the minimum-latency paths to each view. Routing messages can be piggy-backed on regular or NothingToSend messages as in the case of energy-based routes. Local repairs can be performed as in the case of energy-based routing, but by considering latency as the primary metric for evaluating the goodness of a route.

## 4 Experimental Evaluation

We implemented a prototype of wave scheduling in the NS-2 Network Simulator [16] and compared its performance

Average Message Delay Evaluated From Routing Tables



Figure 3: Delay vs energy routing

with other approaches. In Section 4.1, we test the behaviour of wave schedules under different routing metrics, as well as varying the number of views and empty cells. Section 4.2 presents the performance of two competing tree-based scheduling approaches and Section 4.3 shows the behavior of IEEE 802.11 with various duty cycles. A comparison of wave schedules with the other approaches is presented in section 4.4.

#### 4.1 Wave Scheduling

We simulate a network of 20 by 20 grid cells of size  $100m^2$  each. The ratio of interference to communication range is 550/250 and the ratios between radio idle, receive and transmit power are 1:1.2:1.6. Every edge activation between two consecutive cells lasts for 200ms. In the wave schedules, all routing happens at the level of the grid overlay network. A node can send about 10 packets during an edge activation given a link bandwidth of 20kbps. The receiver wakes up 30ms before the sender to avoid message loss when clocks are subject to small drifts.

The size of a square in a pipelined wave is set to 8 by 8 grid cells (instead of 7 by 7) in order to avoid interference as a result of small synchronization errors. Experiments run for 1000 seconds and the traffic workload varies from 0 to 2500 messages. The time that a message is generated is selected at random, uniformly over the simulation period. The source location of a message is randomly selected to be any of the non-empty cells, and the destination to be any of the views. Cells containing views and empty cells are randomly distributed in the network.

**Energy- vs. delay-based routing**. We first compare the behavior of the PipelinedWave schedule under two wave routing metrics: minimum hop-count and minimum-delay. Recall from Section 3.2 that due to the scheduling of the waves, the path with minimum delay is not necessarily the path of minimum hop count. Figure 3 shows the average path delay under light load for the two metrics, i.e. it shows the time between generation of a message at its source and delivery of the message at its destination. This delay is computed by deriving information from the routing tables of the nodes. It coincides with the real message propagation delay when the traffic is low and nodes can completely drain their buffer during an edge activation. The minimum-energy routing metric defines paths



Figure 4: Effect of views on delay



Figure 5: Effect of views on energy

with higher delay than the minimum-delay metric and the gap increases as we increase the number of holes from 0 to 100 (25% of all cells). For 100 holes (or empty cells), the minimum-energy metric yields paths that are 30% slower than the minimum-delay metric. The energy overhead of the minimum-delay metric was observed to be negligible. In the remainder of the section, we use minimum-delay as the default routing metric.

Scalability with the number of views. Our second experiment shows the scalability of our scheme with respect to the number of view nodes. Figure 4 shows the average observed message delay, which captures queueing delay due to traffic. We set the number of empty cells to be 0. With more view nodes, the load is better balanced across the network, the average message propagation delay is smaller and the overall capacity of the network increases. With more than 200 messages for a single view the network is overloaded, and the queues in the network start to grow, and they would continue to grow without bounds if we would not have limited the length of the experiment to 1000 seconds. Figure 5 shows that the energy usage of the wave does not increase with the number of views, for a given This confirms the nice behavior number of messages. of wave routing which makes it exceptionally suitable for sensor networks with multiple *gateway* (or *view*) nodes.

Effect of empty cells. We also examine the impact of

#### Average Message Delay



Figure 6: Effect of holes on delay



Figure 7: Effect of holes on energy

empty cells on the performance of wave schedules. The number of views is 10 and a randomly selected set of 0 to 80 cells are set to be holes. Figure 6 shows that the message latency increases with the number of holes: messages wait longer in order to make a turn to bypass a hole. The capacity of the network is only 500 messages for 20% (80) holes (the message delay increases considerably after that point), whereas it rises to more than 1500 for networks without holes. Interestingly, the average energy consumption per non-empty cell (per node) increases with the number of empty cells, as shown in Figure 7. Although fewer messages are delivered per time unit, these messages follow longer paths. Thus every node ends up routing more messages and spending more energy.

#### 4.2 Tree Scheduling

We compare wave scheduling with an existing tree-based scheduling and routing scheme [13]. Trees are generated using a flooding mechanism initiated at each view node. Every node selects as its parent the neighbor on the shortest path to the root (view). It is therefore expected that the paths used in tree schedules are shorter than paths used in waves, since the latter are built on top of the grid overlay. Routing in a tree is trivial: each non-view node forwards every message it receives to its parent. In a tree-



Figure 8: Delay: consecutive trees



Figure 9: Energy: consecutive trees

based schedule, we activate edges in reverse order of their distance from the root, enabling a message to propagate from any leaf of the tree to the view node in a single tree activation period. Every tree edge is activated for 200ms, as in the case of the wave.

To generalize tree scheduling to handle multiple views, we construct a collection of spanning trees, one tree rooted at each view server. An edge activation schedule can then be derived in several ways. At one extreme is a *conservative* schedule, which is simply a concatenation of schedules for the individual trees. The simplest conservative schedule is to activate tree rooted at view i + 1 immediately after all edges of tree rooted at view i have been activated. In this simple conservative schedule, latency grows linearly with the number of views. In our experiments we study energy-efficient variants of this simple schedule: We define a period p of repeating the activation of every tree. If we have *m* views, the first tree is activated at times  $\{0, p, \ldots\}$ , the second at  $\{p/m, p + p/m, \ldots\}$ , and so on. We assume that the interval p/m is long enough to activate all edges of a single tree, so that consecutive activations do not overlap. In Figures 8 and 9, these schedules are referred to as  $Tag\_Consec\_Every\_p$ , where p is the period between two activations of the same tree.

At the other extreme, we consider *aggressive* schedules that activate all trees in parallel. In the simplest aggres-



Figure 10: Delay: parallel trees



Figure 11: Energy: parallel trees

sive schedule, which is called  $Tag\_Parall$ , consecutive activations of the same tree follow one another immediately after completion. In order to study power-saving variants of the aggressive schedules, we consider periodic activations of the same tree. In our experiments, we use the name  $Tag\_Parall\_Everg\_p$  to refer to aggressive schedules in which all trees are activated concurrently every p seconds (Figures 10 and 11).

In both consecutive and parallel schedules, we observe a graceful tradeoff between energy and delay. As the activation period increases, the energy decreases at the expense of higher message latency and smaller network capacity. Applications aiming at energy preservation should take into consideration the traffic load in order to determine an energy-efficient tree schedule. For instance, the most energy-efficient consecutive schedule that achieves a capacity of 1000 messages has period 60 seconds (Figure 8). Likewise, the most energy-efficient parallel schedule that achieves a capacity of 1000 messages is activated approximately every 12 seconds (Figure 10). Beyond 1000 messages (per 1000 seconds), the delay for these two schedules starts increasing and it would increase without bounds had we continued to generate messages with the same rate for longer periods.



Figure 13: Energy: 802.11

#### 4.3 IEEE 802.11 with Different Duty Cycles

Besides tree scheduling, in which edges are activated in reverse order of their distance to the root, we also study power-conserving variants of the IEEE 802.11 protocol. We vary the duty cycle of the protocol, by turning off the radio regularly and allowing communication only 1 to 10%of the time. The performance of the resulting schemes, named Duty\_Cycle\_x, is shown in Figures 12 and 13. Routing is performed as in tree-scheduling, i.e. messages follow the shortest paths to the views. Notice that for a load of 1000 messages we can only select duty cycles greater than 8%, otherwise the traffic exceeds network capacity and the queues increase without bound. The reader can see trends in energy and delay similar to those observed in the treescheduling schemes. As the duty cycle decreaases, the average message delay decreases significantly at the expense of higher energy usage.

## 4.4 Comparison with Other Schemes

In order to compare different protocols we first select a traffic load and then consider only protocols that can serve this load without exceeding capacity (the point at which average delay begins to increase). We compare the most energy-efficient versions of different pro-



Figure 14: Comparing schemes



Figure 15: Comparing schemes

tocols (with 10 views and 10% empty cells): for 1000 messages, we select the variants *Tag\_Consec\_Every\_60*, *Tag\_Parall\_Every\_12*, *Duty\_Cycle\_8* and the pipelined wave with step 1. From the previous graphs, the reader can see that these are the variants of different protocol-s that accomodate the given traffic with the least energy consumption.

Figure 14 shows that the wave protocol has the longest delay, followed by the consecutive tree schedule, the parallel tree schedule and the 802.11 (with duty cycle 8%). The reverse pattern is observed with respect to node energy consumption in Figure 15. The wave protocol is at one extreme, offering the most energy savings (better by an order of magnitude than any other scheme) at the cost of higher delay. The 802.11 protocol with duty cycle 8% is at the other extreme offering very small message delays at the cost of higher energy. The energy-delay tradeoff of the two tree scheduling algorithms is also worth observing: activating trees consecutively (as opposed to concurrently) saves energy because it avoids interference among different trees, but it incurs higher message latencies.

## 5 Related Work

The advent of sensor network technology has recently attracted a lot of attention to MAC and routing protocols that are specifically tailored for energy-constrained ad-hoc wireless systems.

MAC protocols: Medium access protocols are divided into two main categories, contention-based and schedulebased protocols, depending on whether they resolve or completely avoid collisions at the wireless medium. IEEE 802.11 [17] is the most widely used contention-based protocol; although nodes can periodically switch to a power saving mode, in the active periods they suffer from interference and overhearing. The PAMAS MAC-level protocol turns radios off when nodes are not communicating [18], but it requires a second channel for RTS-CTS messages. PicoNet also allows nodes to turn off their radios [19]; a node wishing to communicate must stay awake listening for a broadcast message announcing its neighbor's reactivation. In S-MAC [20, 21], nodes are locally synchronized to follow a periodic listen and sleep scheme. S-MAC does not explicitly avoid contention for the medium, but reduces the period of overhearing by sending long DATA packets annotated with their lengths. Sift [22] is a randomized C-SMA protocol that aims at reducing latency, rather than energy, in case of spatially-correlated contention.

Schedule-based MAC protocols conserve energy by avoiding message retransmissions or idle listening [23, 24, 25]. NAMA [24] and TRAMA [25] avoid all collisions at the MAC layer by announcing the schedules of nodes in the 2-hop neighborhood and electing nodes to transmit in a given time slot. Our waves avoid schedule propagation overhead, at the expense of having fixed slots for every edge activation. Fixed assignment of communication slots affects message latency, but not the energy consumption at the nodes. TRAMA does not consider interference due to ACK messages, since it assumes that nodes that are three hops away can schedule transmissions cuncurrently.

GAF (Geographical Adaptive Fidelity) [8, 26] is a topology control scheme that conserves energy by identifying nodes that are equivalent from a routing perspective (belong to the same cell) and then turning off unnecessary nodes. The proposed wave algorithms are tightly integrated with the GAF protocol. Unlike S-MAC (a contentionbased scheme) and TRAMA (a schedule-based scheme), under low traffic, the propagation delay of messages from a source to a destination over a multi-hop path is almost *constant*. It depends only on the topology of the network, i.e. which cells are empty, which does not change very rapidly. This desirable property stems from the fact that wave schedules coordinate radio usage across the sensor network.

**Routing algorithms:** Several routing protocols for adhoc networks have been proposed in the literature [27]. There has also been a plethora of work on energy-aware routing [18, 28, 29] but without considering the interplay of routing and scheduling. The TinyDB Project at Berkeley investigates tree-based routing and scheduling techniques for sensor networks [13, 30]. Tree-based routing is tightly combined with node scheduling; all nodes in the same level of the tree are scheduled to send messages to their parents concurrently at a time interval that depends on their distance from the root. Tree-based routing and scheduling is a representative example of the tight coupling between the MAC and routing layers in sensor networks. In this paper we have shown a different kind of interaction, namely how given a certain schedule of edge activations, we can identify

routes that yield minimum message delays.

An energy-efficient aggregation tree using data-centric reinforcement strategies is proposed in [31]. A two-tier approach for data dissemination to multiple mobile sinks is discussed in [32]. Pearlman et al. [33] propose an energy dependent participation scheme, where a node periodically re-evaluates its participation in the network based on the residual energy in its battery. GEAR [29] uses energyaware neighbor selection to route a packet towards a target region and restricted flooding to disseminate the packet inside the destination region; it addresses the problem of energy conservation from a routing perspective without considering the interplay of routing and node scheduling.

## 6 Conclusions and Future Work

In this paper, we have presented a class of algorithms that allow us to trade energy versus delay for data dissemination in sensor networks. Our approach is based on carefully *scheduling* the sensor nodes so that each node can stay idle most of the time, turning on its radio only at scheduled intervals during which it can receive or send a message. Our experiments show that the proposed wave scheduling algorithm results in significant energy savings at the expense of increased message latency.

In the future, we plan to study irregular wave schedules, in which we relax the current assumption that every directed edge in the network is scheduled regularly once per period, and thus has the same capacity. In practice, incoming edges to view nodes are expected to be more heavily loaded than edges at the border of the network. We believe that better network utilization can be achieved by considering a more general class of wave schedules in which different edges are activated with different rates. For instance, the network can be divided into highways (frequently-activated edges) and driveways (low-capacity edges). It would be interesting to study the tradeoff between energy and delay in such an irregular model.

Another interesting direction is to investigate the problem of time synchronization for wave schedules. Existing approaches, like RBS [11] and TPSN [12], provide tight synchronization bounds and exhibit good multi-hop behavior – with high probability, the error is less than linear in the number of hops. Using a tree-based approach, they aim at providing a global timescale exceeding the more relaxed requirements of wave schedules. Their performance is therefore bound to decay for very large networks. We intend to investigate highly distributed and scalable algorithms that are specifically tailored to achieve good time synchronization among nodes within interference range, instead of achieving global synchronization.

## References

- S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, and S. Shenker, "Ght: A geographic hash table for data-centric storage," in WSNA, 2002.
- [2] A. Ghose, J. Grossklags, and J. Chuang, "Resilient data-centric storage in wireless ad-hoc sensor networks," in *MDM*, 2003, pp. 45–62.
- [3] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris, "Span: A energy-efficient coordination algorithm

for topology maintenance in ad hoc wireless networks," *ACM Wireless Networks*, vol. 8, no. 5, September 2002.

- [4] O. Kasten, "Energy consumption," Tech. Rep., ETH-Zurich, 2001.
- [5] M. Stemm and R. Katz, "Measuring and reducing energy consumption of network interfaces in hand-held devices," *IEICE Transactions on Communications*, vol. E80-B, pp. 1125–1131, 1997.
- [6] N. Trigoni, Y. Yao, A. Demers, J. Gehrke and R. Rajaraman, "WaveScheduling: Energy-Efficient Data Dissemination for Sensor Networks," 2004 cougar.cs.cornell.edu.
- [7] A. Sen and M. Huson, "A new model for scheduling packet radio networks," in *INFOCOM*, 1996, pp. 1116–1124.
- [8] Y. Xu, J. Heidemann, and D Estrin, "Geographyinformed energy conservation for ad hoc routing," in *MOBICOM*, 2001, pp. 70–84.
- [9] P Bahl and V.N. Padmanabhan, "RADAR: An inbuilding RF-based user location and tracking system," in *INFOCOM (2)*, 2000, pp. 775–784.
- [10] N. Bulusu, J. Heidemann, and D. Estrin, "Gps-less low cost outdoor localization for very small devices," 2000.
- [11] J. Elson, L. Girod, and D. Estrin, "Fine-grained network time synchronization using reference broadcasts," ACM SIGOPS Operating Systems Review, SI: Physical Interface, vol. 36, pp. 147-163, 2002.
- [12] S. Ganeriwal, R. Kumar, and M.B. Srivastava, "Timing-sync protocol for sensor networks," in *SEN-SYS*, 2003, pp. 138–149.
- [13] Samuel R. Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong, "Tag: A tiny aggregation service for ad-hoc sensor networks," in OSDI, 2002.
- [14] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia, "Routing with guaranteed delivery in ad hoc wireless networks," *Wireless Networks*, vol. 7, no. 6, pp. 609– 616, 2001.
- [15] B. Karp and H.T. Kung, "GPSR: greedy perimeter stateless routing for wireless networks," in *MOBICOM*, 2000, pp. 243–254.
- [16] L. Breslau, D. Estrin, K. Fall, S. Floyd, J. Heidemann, A. Helmy, P. Huang, S. McCanne, K. Varadhan, Y. Xu, and H. Yu, "Advances in network simulation," *IEEE Computer*, vol. 33, no. 5, pp. 59-67, May 2000.
- [17] IEEE Computer Society, "Wireless LAN medium access control (mac) and physical layer specification," IEEE Std 802.11, 1999.
- [18] S. Singh, M. Woo, and C.S. Raghavendra, "Poweraware routing in mobile ad hoc networks," ACM SIG-MOBILE, 1998, pp. 181–190, ACM Press.
- [19] F. Bennett, D. Clarke, J. Evans, A. Hopper, A. Jones, and D. Leask, "Piconet: Embedded Mobile Networking," *IEEE Personal Communications*, vol. 4, no. 5, pp. 8-15, Oct. 1997.

- [20] W. Ye, J. Heidemann, and D. Estrin, "An energyefficient MAC protocol for wireless sensor networks," in *INFOCOM*, 2002, pp. 1567–1576.
- [21] W. Ye, J. Heidemann, and D. Estrin, "Medium access control with coordinated, adaptive sleeping for wireless sensor networks," Tech. Rep. ISI-TR-567, USC/Information Sciences Institute, January 2003.
- [22] K. Jamieson, H. Balakrishnan, and Y.C. Tay, "Sift: A mac protocol for event-driven wireless sensor networks," Tech. Rep., MIT, May 2003.
- [23] G. J. Pottie and W. J. Kaiser, "Embedding the Internet: wireless integrated network sensors," *Communications of the ACM*, vol. 43, no. 5, pp. 51–51, May 2000.
- [24] L. Bao and J.J. Garcia-Luna-Aceves, "A new approach to channel access scheduling for ad hoc networks," in *MOBICOM*, 2001, pp. 210–221.
- [25] V. Rajendran, K. Obraczka, and J.J. Garcia-Luna-Aceves, "Energy-efficient collision-free medium access control for wireless sensor networks," in *SENSYS*, 2003, pp. 181–192.
- [26] Y. Xu, S. Bien, Y. Mori, J. Heidemann, and D. Estrin, "Topology control protocols to conserve energy inwireless ad hoc networks," Tech. Rep. 6, University of California, Los Angeles, Center for Embedded Networked Computing, January 2003, submitted for publication.
- [27] C.E. Perkins, Ad hoc networking, Addison-Wesley Longman Publishing Co., Inc., 2001.
- [28] J.-H. Chang and L. Tassiulas, "Energy conserving routing in wireless ad-hoc networks," in *INFOCOM*, 2000, pp. 22–31.
- [29] Y. Yu, R. Govindan, and D. Estrin, "Geographical and energy aware routing: A recursive data dissemination protocol for wireless sensor networks," Tech. Rep. UCLA/CSD-TR-01-0023, University of Southern California, May 2001.
- [30] J.M. Hellerstein, W. Hong, S. Madden, and K. Stanek, "Beyond average: Towards sophisticated sensing with queries," in *IPSN*, 2003.
- [31] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion: A scalable and robust communication paradigm for sensor networks," ACM SIGMO-BILE, 2000, pp. 56–67, ACM Press.
- [32] F. Ye, H. Luo, J. Cheng, S. Lu, and L. Zhang, "A twotier data dissemination model for large-scale wireless sensor networks," in *MOBICOM*, 2002.
- [33] M.R. Pearlman, J. Deng, B. Liang, and Z.J. Haas, "Elective participation in ad hoc networks based on energy consumption," in *IEEE GLOBECOM*, 2002, pp. 17–21.