WaveScheduling: Energy-Efficient Data Dissemination for Sensor Networks

Niki Trigoni[1] Yong Yao[1] Alan Demers[1] Johannes Gehrke[1]
Rajmohan Rajaraman[2]

[1]Department of Computer Science, Cornell University
[2]College of Computer and Information Science, Northeastern University

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 well-defined 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 SensorDBMS 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 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.

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 power-efficient 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 off their radios most of the time. Since current generation radios consume nearly as much power when listening or receiving as when transmitting [3, 4, 5], the ability to turn them off when not needed yields significant energy savings. (The idle:receive:transmit ratios observed in these studies are 1:1.2:1.7 [3], 1:2:2.5 [4], and 1:1.05:1.4 [5].)

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 *schedul-*

Proceedings of the 30th VLDB Conference, Toronto, Canada, 2004

ing algorithm that activates network edges such that their transmissions do not interfere with one another and a *routing algorithm* for selecting routes for individual messages. Two important performance metrics are *energy consumption* and *latency*. In this section, we sketch complexity results for the following optimization problems with respect to both energy consumption and latency metrics: (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. The proofs of the following complexity results can be found in [6].

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 α is a *fixed start-up cost* for turning the radio on, β 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 energy-optimal routes given an activation schedule can be solved in polynomial time.*

Latency minimization. Given a communication workload, we seek a data dissemination protocol that minimizes average message propagation latency.

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 present *wave scheduling*, a class of periodic edge activation schedules, and study the close interaction between scheduling and routing with respect to the energy and delay metric. Our scheduling mechanism is layered on top of a protocol like GAF [7], 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 [8, 9]. 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 maximal transmission range of a node. The proposed wave schedules leverage the abstraction of partitioning irregularly positioned nodes into cells organized in a rectilinear grid; they focus on reducing energy consumption by coordinating inter-cell communication. For simplicity, we assume a square rectilinear grid of $N \times N$ nodes. Cell $(0, 0)$ is located at the southwest corner of the network. Cells $(i + 1, j)$, $(i, j + 1)$, $(i - 1, j)$, and $(i, j - 1)$ are the east, north, west, and south neighbors, respectively, of cell (i, j).

3.1 Wave Scheduling

Edge activation. In our *wave schedules*, every (directed) edge of the rectilinear grid is activated periodically at well-defined communication intervals, called *send-receive* intervals. The interval between activations is the same for all edges and is referred to as the *period*. An edge activation $A \rightarrow B$ consists of a *contention-based* and a *collision-free* period. During the contention-based period, all nodes within cell A turn on their radios in order to run the GAF protocol. If the old leader is energy-drained, a re-election protocol selects a new leader and state (routing table and message queue) is transferred to the new leader. The remaining nodes then send all readings generated since the previous GAF period to the new leader. This adapted version of the GAF protocol avoids interference caused by concurrent leader election in nearby cells. In the collision-free period leaders of A and B turn on their radios preparing for inter-cell communication. If A has no data messages to send, it sends a special *NothingToSend* (NTS) message, which allows both nodes to turn off their radios before the end of the allotted interval. The node duty cycle is thus adjusted to local traffic. In the collision-free period a data (or NTS) message is not preceded by a pair of RTS-CTS messages, but simply followed by an ACK.

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 Δ 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, two cells (i, j) and (i_1, j) can transmit simultaneously if $i - i_1 \geq \lceil \Delta \cdot R/L \rceil + 1$, which we denote by g. In the SimpleWave schedule, we schedule together edges that are g positions apart. Figure 1 illustrates the SimpleWave schedule on a 10×10 network, with cell size $L = 100m$, yielding a g of 7. The north phase starts at time 1 and lasts for 51 send-receive intervals during which every north edge is activated exactly once. The following east phase starts at time 52. 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) it no longer interferes with node communication in the first two columns. Notice that at time 59 one can schedule four edges concurrently: $(7, 0) \rightarrow (8, 0)$,

$(7,7) \rightarrow (8,7)$, $(0,1) \rightarrow (1,1)$ and $(0,8) \rightarrow (1,8)$.

In a *SimpleWave*, each phase takes $(N-1)+(g-1)g$ send-receive intervals and the entire wave period lasts for $4((N-1)+(g-1)g)$ intervals. This prevents the distributed deployment of the algorithm in a dynamic network: when a new cell joins (or leaves) the network, it affects the wave period and therefore the activation times of all the other supernodes. Furthermore, every node needs to know the size of the network. Another important downside of the *SimpleWave* algorithm is that it underutilizes the capacity of the network.

PipelinedWave. This algorithm is motivated by the need for distributed and scalable schedules that make good use of network capacity. Conceptually, a network can be divided in a number of small fixed-size $(g \times g)$ squares, where all squares have exactly the same schedule. In such a network, the schedule of an edge is determined by its relative location in the square. Since all edges within the same square interfere with one another, we can schedule at most one edge at a time. The period of the resulting schedule is $4g^2$ send-receive intervals. Two edges are scheduled concurrently if they have the same direction and the sender nodes have exactly the same local coordinates within a $g \times g$ square. The *PipelinedWave* schedules a maximum number of non-interfering edges at each send-receive interval.

The *PipelinedWave* algorithm has two important properties: i) it is easily deployable in a distributed manner, since local coordination suffices for scheduling a new cell and ii) it is scalable, because node schedules are not affected by the size of the network. When a cell gets newly occupied, the associated node waits for at most one period in order to interact with its neighbors and determine its local coordinates. By overhearing the schedules of its immediate neighbors it easily determines its own schedule. When a node enters or leaves the network, the schedules of the remaining cells do not change.

A modified version of the *PipelinedWave* algorithm does not define identical schedules for each square, but shifts schedules by g positions with respect to the schedules of the four neighbor squares. More specifically, the east wave of a square is shifted g send-receive intervals earlier than the east wave of the west neighbor square, the north wave is shifted g 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 new algorithm (which is the one tested in section 4) decreases the latency of message delivery at the square boundaries. 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 has no noticeable impact on the performance of the wave schedule [6].

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 g 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 ϵ, nodes that are g positions away from each other are synchronized within $g\epsilon$. In every edge activation, we schedule the receiver to turn on the radio ϵ time units earlier than the scheduled time according to its local clock. Recently proposed synchronization protocols for sensor networks (e.g., RBS [10] and TPSN [11]) provide tight synchronization bounds (e.g., $0.02ms$ for neighbor nodes [11]) and exhibit a nice multi-hop behavior. Their performance is bound to decay for very large networks, in which case we assume that a few GPS-equipped nodes will undertake the synchronization task for the local region.

3.2 Routing

The proposed wave schedules are TDMA-based MAC protocols that assign periodic transmission slots to inter-cell. 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*. An interesting outcome of our study is 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. Each node in the network maintains a small in-memory routing table of size proportional to the number of view servers. For each view server, it 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 in [12]. 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 [13, 14]. 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 C also 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 $N^* \rightarrow C$ remains at C before being forwarded with the next wave via $C \rightarrow N$ 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 con-

verges, every node has determined the minimum-latency paths to each view. Routing messages can be piggybacked on regular or NothingToSend messages as in the case of energy-based routes.

4 Experimental Evaluation

We implemented a prototype of wave scheduling in the NS-2 Network Simulator [15] and compared its performance with two other approaches: (i) an existing tree-based scheduling and routing scheme [12] and (ii) using IEEE 802.11 with different duty cycles.

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. 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 in order to allow for clock drift. The size of a square in a pipelined wave is set to 8 by 8 grid cells. 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.

We first compare the behavior of the PipelinedWave schedule under two wave routing metrics: the minimum hop-count and the minimum-delay path. Figure 3 shows the average path delay, under light load, for the two metrics, i.e. the time between a generation of a message at a source and its delivery at the destination. The minimum-energy routing metric defines paths 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). The energy overhead of the minimum-delay metric was observed to be negligible.

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. Figure 5 shows that the energy usage of the wave does not increase with the number of views, for a given number of messages.

We also examine the impact of 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 and every node ends up routing a higher number of messages, therefore spending more energy.

Tree Scheduling We compare wave scheduling with an existing tree-based scheduling and routing scheme [12]. Trees are generated as a result of 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. Routing in a tree is trivial: each non-view node forwards every message it receives to its parent. In a tree-based schedule, we activate edges in reverse order of their distance from the root. Every tree edge is activated for 200ms seconds, 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. 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 our experiments, we use the name $Tag\_Parall\_Every\_p$ to refer to aggressive schedules in which all trees are activated concurrently every p seconds. For instance, we activate all m trees together at times $\{0, p, 2p, \ldots\}$. Figures 10 and 11 show a graceful tradeoff between energy and delay as we increase the length of period p. We note that the most energy-efficient consecutive schedule that achieves a capacity of 1000 messages has period 60 seconds. Likewise, the most energy-efficient parallel schedule that achieves a capacity of 1000 messages is activated approximately every 12 seconds. 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.

IEEE 802.11 with different duty cycles. 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 during 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 tree-scheduling schemes.

Comparison of waves with other schemes. In

order to compare different protocols we first select a given traffic load and we only consider protocols that can serve this load without exceeding capacity, which is the point at which average delay starts increasing. In fact, we compare the most energy-efficient versions of different protocols (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. 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 higher 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 adhoc wireless systems.

MAC protocols: IEEE 802.11 [16] 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 [17], but it requires a second channel for RTS-CTS messages. PicoNet also allows nodes to turn off their radios [18]; a node wishing to communicate must stay awake listening for a broadcast message announcing its neighbor's reactivation. In S-MAC [19, 20], 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. NAMA and TRAMA 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.

Routing algorithms: Several routing protocols for ad-hoc networks have been proposed in the literature [21, 22, 23, 24, 25]. There has also been a plethora of work on energy-aware routing [26, 17, 32] but without considering the interplay of routing and scheduling. The TinyDB Project at Berkeley investigates tree-based routing and scheduling techniques for sensor networks [12, 29]. An energy-efficient aggregation tree using data-centric reinforcement strategies is proposed in [30]. A two-tier approach for data dissemination to multiple mobile sinks is discussed in [31].

6 Conclusions and Future Work

In this paper, we have shown a class of algorithms that allows us to trade energy versus delay for data dissemination in sensor networks. Our approach is based on carefully *scheduling* the sensor nodes such that each node can stay idle most of the time, and only turns on its radio at scheduled intervals during its turn to either receive or send a message. Our experiments show that the proposed wave scheduling algorithm results in significant energy savings at modest increases in latency.

References

[1] 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, Sept 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," www.cs.cornell.edu/∼niki/WaveScheduling.pdf, 2004.

[7] Y. Xu, J. Heidemann, and D. Estrin, "Geography-informed energy conservation for ad hoc routing," in *MOBICOM*, 2001, pp. 70–84.

[8] P. Bahl and V. N. Padmanabhan, "RADAR: An in-building RF-based user location and tracking system," in *INFOCOM*, 2000, pp. 775–784.

[9] N. Bulusu, J. Heidemann, and D. Estrin, "Gps-less low cost outdoor localization for very small devices," 2000.

[10] J. Elson, L. Girod and D. Estrin, "Fine-Grained Network Time Synchronization using Reference Broadcasts," in *OSDI*, 2002.

[11] S. Ganeriwal, R. Kumar and M. Srivastava, "Timing-sync protocol for sensor networks," in *Sensys*, pp.138–149, 2003.

[12] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "Tag: A tiny aggregation service for ad-hoc sensor networks," in *OSDI*, 2002.

[13] 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.

[14] B. Karp and H. T. Kung, "GPSR: greedy perimeter stateless routing for wireless networks," in *MOBICOM*, pp. 243–254, 2000.

[15] 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, 2000.

[16] IEEE Computer Society, "Wireless LAN medium access control (mac) and physical layer specification," IEEE Std 802.11, 1999.

[17] S. Singh, M. Woo, and C. S. Raghavendra, "Power-aware routing in mobile ad hoc networks," in *MOBICOM*, pp. 181–190, 1998.

[18] 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.

[19] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient MAC protocol for wireless sensor networks," in *INFOCOM*,pp. 1567–1576, 2002.

[20] 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.

[21] C. Perkins and P. Bhagwat, "Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers," in *SIGCOMM*, pp. 234–244, 1994.

[22] D. B Johnson and D. A Maltz, "Dynamic source routing in ad hoc wireless networks," in *Mobile Computing*, vol. 353 of *The Kluwer International Series in Engineering and Computer Science*, 1996.

[23] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva, "A performance comparison of multi-hop wireless ad hoc network routing protocols," in *MOBICOM*, pp. 85–97, 1998.

[24] C. Perkins, "Ad hoc on demand distance vector (aodv) routing," http://citeseer.ist.psu.edu/article/perkins99ad.html, 1999.

[25] V. Park and S. Corson, "Temporally-ordered routing algorithm (tora) version 1 functional specication.," 1999 http://www.ietf.org/internet-drafts/draft-ietf-manet-tora-spec-02.txt.

[26] J.-H. Chang and L. Tassiulas, "Energy conserving routing in wireless ad-hoc networks," in *INFOCOM*, pp. 22–31, 2000.

[27] W. R. Heinzelman, J. Kulik, and H. Balakrishnan, "Adaptive protocols for information dissemination in wireless sensor networks," in *SIGMOBILE*, pp. 174–185, 1999.

[28] S. Madden and M. J. Franklin, "Fjording the stream: An architecture for queries over streaming sensor data," in *ICDE*, 2002.

[29] J. M. Hellerstein, W. Hong, S. Madden, and K. Stanek, "Beyond average: Towards sophisticated sensing with queries," in *IPSN*, 2003.

[30] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion: A scalable and robust communication paradigm for sensor networks," in *SIGMOBILE*, pp. 56–67, 2000.

[31] F. Ye, H. Luo, J. Cheng, S. Lu, and L. Zhang, "A two-tier data dissemination model for large-scale wireless sensor networks," in *MOBICOM*, 2002.

[32] 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, May 2001.

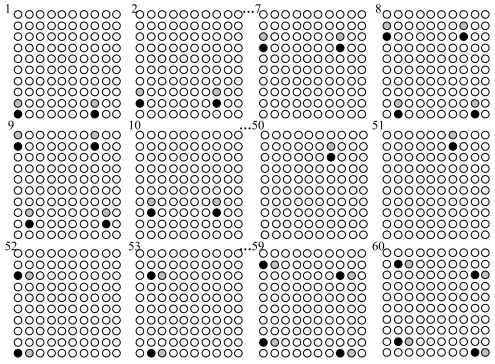

Figure 1: SimpleWave

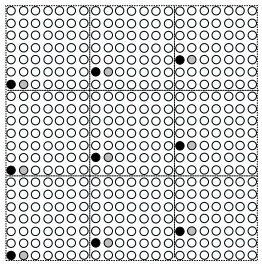

Figure 2: PipelinedWave

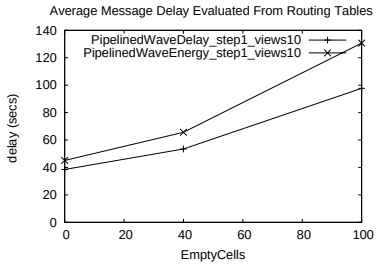

Figure 3: Delay vs energy routing

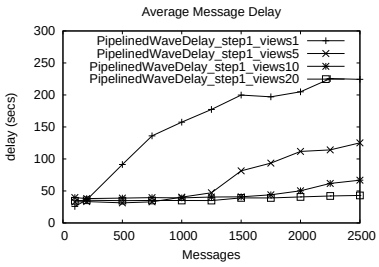

Figure 4: Effect of views on delay

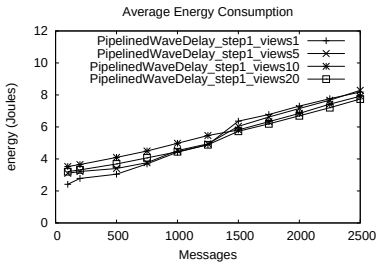

Figure 5: Effect of views on energy

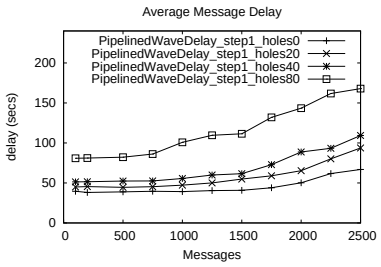

Figure 6: Effect of holes on delay

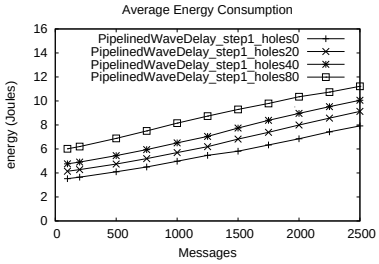

Figure 7: Effect of holes on energy

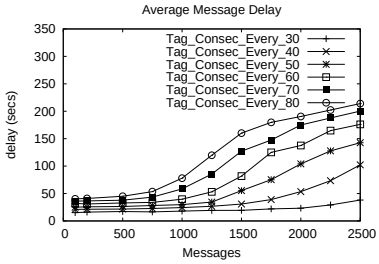

Figure 8: Delay: consecutive trees

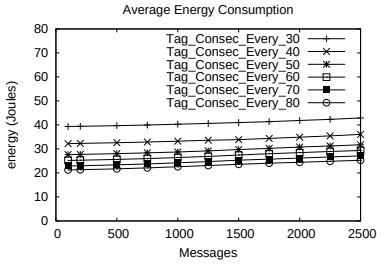

Figure 9: Energy: consecutive trees

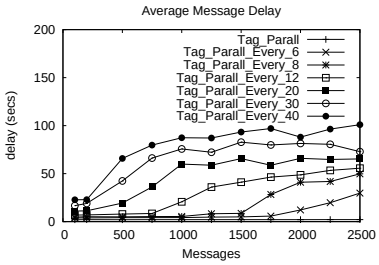

Figure 10: Delay: parallel trees

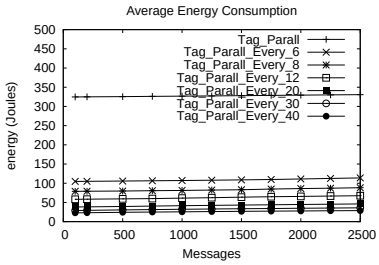

Figure 11: Energy: parallel trees

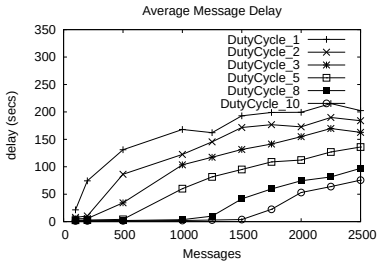

Figure 12: Delay: 802.11

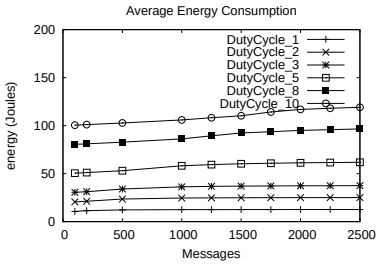

Figure 13: Energy: 802.11

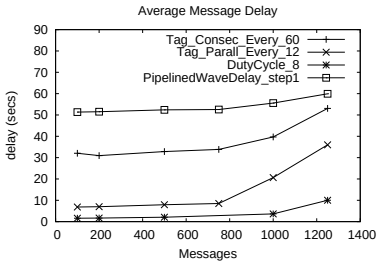

Figure 14: Comparing schemes

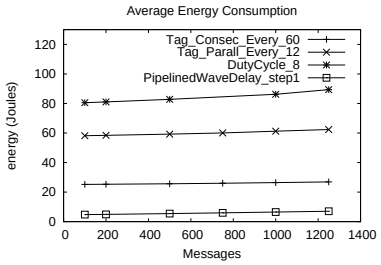

Figure 15: Comparing schemes