

Adaptive Sleep Discipline for Energy Conservation and Robustness in Dense Sensor Networks

Jana van Greunen, Dragan Petrović, Alvis Bonivento,
Jan Rabaey, Kannan Ramchandran, Alberto Sangiovanni-Vincentelli
Department of Electrical Engineering and Computer Sciences
University of California, Berkeley
Berkeley, USA

Abstract— We present an adaptive approach for conserving energy in high-density sensor networks. The proposed method allows sensor nodes to sleep while ensuring that application performance constraints are met. Unlike deterministic algorithms that assume static connectivity, the approach uses a randomized algorithm to provide robustness to the variations in network connectivity. These variations are due to fading channels, depletion or addition of nodes, node mobility, and the sleeping of nodes. The algorithm developed in the paper is extremely lightweight and does not require nodes to keep any state information about their individual neighbors. Based on local observations, each node independently decides when to sleep and wakeup. The algorithm also ensures an evenly distributed workload among nodes, and achieves energy savings proportional to the density of nodes.

Keywords- *Sensor networks, Adaptive systems, Distributed decision-making, Energy conservation*

I. INTRODUCTION

Sensor networks are being developed to serve in a multitude of environmental monitoring and control applications [1]-[3]. Due to the small size and extreme power constraints on sensor nodes, it is desirable to use energy-aware protocols for inter-node communication. The most important way to save energy in a sensor network is to power-down (put to sleep) any node that is not performing useful work. Because sensor nodes are often densely deployed (i.e., to support high-resolution sampling of the environment [4]), there exists a high degree of redundancy in the network topology. Thus, it is possible to design a node communication protocol that can exploit this redundancy and allow nodes to minimize energy consumption by sleeping for the maximum amount of time. The paper describes a lightweight, distributed sleeping discipline that meets these goals while ensuring overall network performance requirements (e.g., routing delay) are met.

Another important aspect of the proposed discipline is robustness and adaptation to changing network connectivity. The network topology is time-varying due to the addition of new nodes (birth), energy depletion in others (death), and the mobility of nodes. Furthermore, real-world deployment of sensor networks has revealed that, even without birth, death, and mobility, sensor network connectivity varies over time. It has been shown that fading and multi-path effects cause the quality of the communication channels to fluctuate [5]. Thus, ensuring robustness to changing network connectivity and

provisioning for an adaptive scheme that performs well under various network conditions is a crucial part of the approach taken in this paper.

The task of a sensor network is to perform distributed sensing. Regardless of the specific sensing application, the information gathered is usually time-sensitive. For a specific sensing task there is an application-specified delay constraint on the processing time of the task. The topology (number of nodes, connectivity) and activity (packet load, node activity) affect the ability of the network to meet this delay constraint. Allowing nodes to sleep changes both the topology and activity of the network. Thus, it is necessary to ensure that the network's ability to perform the required tasks is not diminished by allowing nodes to sleep and conserve energy.

In summary, the main contribution of this work is the development of a sleep discipline that:

- Allows nodes to maximize their sleeping time while ensuring that performance constraints (e.g latency) are met.
- Adapts to different network topologies
- Is robust to communication channel/node density variations
- Requires no additional communication overhead, keeps no state about neighbors, and is very lightweight.
- Ensures fairness (even distribution of load)

In order for the sleep discipline to exploit the density of the network effectively, the nodes must be *equivalent* to each other. In other words, the nodes must be functionally homogenous. Due to their dense deployment and geographic proximity, any one of the equivalent nodes is capable of performing the same tasks. An example application called *opportunistic routing* relies on having many equivalent nodes. Opportunistic routing is a modification of geographic routing; a sender forwards packets to a set of equivalent nodes (that are closer to the destination), instead of forwarding packets to the single neighbor that is the closest to the packet destination. Consider the interaction of opportunistic routing and sleeping (shown in Figure 1). The black node is the sender and has a packet to send. The clear nodes in the shaded region are equivalent and can all forward the packet to its final destination or perform the required processing. If all of the equivalent nodes are sleeping, the sending node has to wait until one of

This work was supported in part by NSF grant# ECS-0225534 "Integrated Sensing: Mitigating Bottlenecks and Hotspots in Wireless Sensor Systems" and MARCO "Collaborative Research in the Design, Verification, and Test of Integrated GigaScale Systems: The GigaScale Systems Research Center"

the equivalent nodes wakes up and announces that it is ready to receive and process packets. If the communication channel between the sending node and the node that wakes up happens to be bad at that moment in time, the sending node will not be aware of the wake-up. Thus, the sending node will simply wait until one of the other equivalent nodes wake up. In order to exploit these equivalent nodes for routing applications, the nodes must know their positions. This assumption is made throughout the rest of the paper.

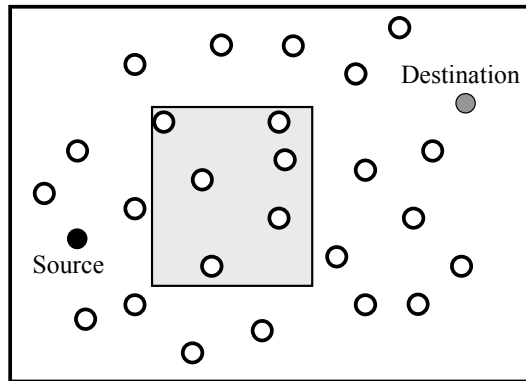


Figure 1. The source node has a packet that needs to be forwarded to the destination. Any one of the nodes in the shaded region can serve as the next hop.

The main goal of the sleeping discipline is minimizing the energy dissipation in the network. There is a trade-off between the amount of time nodes can sleep and the latency of the network. In addition, as the latency of the network increases additional energy is dissipated by sending nodes. These nodes have to remain awake or wake-up periodically until the data has been transmitted. Therefore, it is desirable from an energy perspective to ensure that the candidate receiving nodes wake up often enough to keep this delay low. The amount of time a node sleeps should be dependent on the overall activity of the equivalent nodes.

The goals, constraints and parameters of the proposed approach can be summarized as follows:

A. Goals:

- Exploit density of homogeneous nodes to conserve energy by allowing nodes to turn themselves off;
- Provide robustness to changing network connectivity;
- Minimize sleep discipline's communication overhead.

B. Constraints:

- Network must complete tasks within a certain period (latency constraint)
- Network must be able to carry a certain load (capacity constraint)
- Nodes behave independently, and there is no global notion of time-slots or coordination with neighbors.

- Nodes have no information about individual neighbor's sleeping patterns.

C. Parameters:

- Node wake-up rates

Section II describes related work. The algorithm design is presented in Section III. Section IV sets forth the algorithm. Section V presents the simulation results of the algorithm in a sensing and routing application. Finally, Section VI concludes the work and describes the directions of future research.

II. RELATED WORK

There are several existing algorithms that determine when certain nodes can sleep in order to reduce system-level energy consumption. GAF [6], SPAN [7] and CS [8] focus on controlling the effective network topology by selecting a connected set of nodes to be more active and turning the rest of the nodes off. The GAF algorithm works by identifying redundant nodes (those that are equivalent to active nodes from a routing perspective) and turning these off. SPAN is similar to GAF and attempts to build a backbone for routing by keeping a subset of nodes awake. The SPAN algorithm is distributed and nodes make local decisions whether to join the backbone or to sleep. The decision to join is based on a node's remaining energy reserve and the network benefit resulting from its activity.

The CS algorithm is an approximation to the NP complete problem of finding a dominating set on a graph. Given static connectivity and a dominating set of active nodes, the sensor network will be fully connected. Under CS, nodes potentially require knowledge of the entire network topology because a node needs to verify that there exists a path that visits all of its "awake" neighbors. This path can traverse the entire network.

Unlike the approach taken in this paper, previous approaches require the nodes to keep state about their individual neighbors. This need for timely state information requires additional communication. The existing algorithms also do not consider the effects of time-varying communication channels between nodes.

III. ALGORITHM DESIGN

A distributed design was chosen in order to avoid the communication overhead required to synchronize the sleeping behavior of the nodes. In addition, the design avoids deterministic sleeping behavior because the time-varying nature of the communication channels and network topology makes deterministic schemes unreliable and susceptible to failure. Including randomness in the sleeping algorithm provides robustness.

In order to concretely expound our adaptive sleeping discipline, the scope of the research has been narrowed to delay constrained routing. The remainder of the paper will set forth the algorithm and illustrate its performance in the context of opportunistic routing. The following assumptions are made:

- A node can process and store all packets that are forwarded to it. In other words, the amount of traffic is

low enough that it never exceeds a node's buffer space, or nodes have infinite buffering capacity.

- When a packet is transmitted, the transmitting node appends a timestamp to the packet, specifying how long it has been waiting to transmit the packet.
- The per-hop delay constraint is specified by a tuple $\{\tau, p_\tau\}$ and is met when, with probability p_τ , the hop-delay will be less than τ , *i.e.*, $P(\text{Per-hop delay} > \tau) < p_\tau$. Note: the per-hop delay constraint can be translated into an end-to-end delay constraint if the number of hops and joint distribution of the per-hop delays are known. The expected end-to-end delay is the sum of expected per-hop delays. If one assumes that per-hop delays are independent, the variance of per-hop delays is the sum of the variances of per-hop delays, and the joint distribution is the product of distributions of one-hop delays. The joint distribution can be used to calculate a similar probabilistic delay constraint on the end-to-end delay.

The algorithm is distributed and each node adjusts its sleeping time independently. In order to ensure that the combined activity of equivalent nodes is high enough to meet the delay constraint, each node needs to have an idea of the overall activity in its region. If the packet arrivals are memory-less, the node can estimate how long it has been since the last awakening of one of its equivalent nodes. This estimate is made based on how many packets are waiting to be forwarded when the node wakes up, and how long they have been waiting. If the estimated activity is too low to satisfy the delay constraint, the node decides to wake up more often. Conversely, if the activity is higher than necessary, the node decides to sleep longer.

Note that the packet arrivals are determined by the wake up events of the previous hop nodes. Therefore, having a memory-less node wake-up process ensures that packet arrivals also form a memory-less process. The exponential distribution is the only continuous distribution that is memory-less [9]. Therefore, the node sleeping times are exponentially distributed in the algorithm. Under this scheme the node wake-up process, as well as the packet arrival process, are Poisson processes.¹

Another property of the exponential distribution is that the minimum of N exponential random variables is an exponential random variable with parameter μ , which is equal to the sum of the individual parameters. In the sleeping problem, the distribution of one-hop delay is determined by the distribution of time from the packet arrival until the first equivalent node wakes up. The node wake-ups are all exponential and memory-less. Thus, the per-hop delay is the minimum of N exponential

¹ Technically the packet arrival process is not Poisson. For example, if a node receives k packets from the neighboring block when it wakes up, those k arrivals happen within a short time and are not Poisson. However, this problem can be circumvented by combining the k received packets into a single packet for the next hop. Also, in practice, traffic in the network is relatively sparse and it is unusual for nodes to receive a large number of packets at once.

variables, where N is the number of equivalent nodes. More concisely, the per-hop delay is exponential with parameter μ (μ is the combined wake-up frequency of the equivalent nodes). The per-hop delay depends only on the total (sum) wake-up rate of equivalent nodes and not an individual node's wake-up rate.

Given that the per-hop delay has an exponential distribution, and the delay constraint will be satisfied iff

$$p_\tau \geq \int_{\tau}^{\infty} \mu e^{-\mu x} dx = e^{-\mu\tau} \quad (1)$$

Rearranging (1) $\mu \geq \frac{1}{\tau} \ln \frac{1}{p_\tau}$. This minimum value of μ that satisfies (1), is denoted as μ^* in the rest of the paper. μ^* is the optimal wake-up rate because it minimizes the node wake-up rate while satisfying the delay constraint.

The intuition behind the algorithm is that nodes can pre-compute the optimal wake-up rate, μ^* . Every time a node wakes up, it receives packets that are waiting to be processed by the set of equivalent nodes it belongs to. Based on the arrivals of packets, the node forms an estimate of the aggregate wake up rate of its equivalent nodes, and decides to wake up more/less often depending on whether the estimated wake up rate is less/greater than μ^* .

The estimate of the aggregate wake-up rate for a region is a crucial part of the algorithm. The Poisson nature of the node wake-up and packet arrival processes allows the nodes to form an unbiased estimate of the aggregate wake-up rate without additional communication. Let μ_B denote the wake-up rate of equivalent nodes in a region B . The node wake-up process and packet arrival process for B can be viewed as a combined Poisson process with rate μ_c . The memory-less property of the Poisson process enables the nodes to estimate μ_B from observing arrivals in the combined process. Suppose a node wakes up at time t . Given t_0 (the arrival time of the first packet to be forwarded into region B after all equivalent nodes have gone to sleep) and k (the number of packets that have arrived since time t_0), a node can form an estimate of μ_c .

$$\hat{\mu}_c = \frac{k}{(t - t_0)} \quad (2)$$

Further, the arrival in the combined process prior to the packet arrival at t_0 is a node wake-up. The expected time at which the previous node wake-up occurred is given by $(t_0 - 1/\mu_c)$ which is the last known arrival time minus the expected inter-arrival time in the combined Poisson process. Thus, the node's estimate of the aggregate node inter-arrival time is simply t (its own wake-up time) minus the estimated time of the previous node wake-up. The aggregate rate μ_B is the inverse

of the aggregate node inter-arrival time. Thus, the nodes' estimate of μ_B is given by:

$$\hat{\mu}_B = \frac{1}{\left(t - t_0 - \frac{1}{\hat{\mu}_c}\right)}. \quad (3)$$

When there is sufficient traffic in the network, (3) will result in a reliable estimate of μ_B . However, if the traffic is very sparse, the nodes may not observe packets for long periods of time. In this case, a different method of estimation is required. This paper focuses on the case where there is sufficient traffic in the network. An alternative estimation method is currently being developed.

Another important aspect of the sleeping algorithm is load sharing (fairness) between nodes. In order to achieve fairness, the algorithm must ensure that the nodes' independent adjustments converge to a fair equilibrium where the load is equally distributed between equivalent nodes. This problem of achieving fair wake-up rates (with distributed local decisions based on feedback of aggregate behavior) is equivalent to the problem of sharing bandwidth among competing TCP flows with aggregate congestion feedback. In the bandwidth sharing problem, the goal is to ensure that the total bandwidth usage of all the flows does not exceed the capacity of the network and that each of the competing flows gets an equal share of the total available bandwidth. In the problem studied here, the goal is ensuring that the average time between consecutive node wake ups does not exceed the delay bound and that the average sleeping times of all equivalent nodes are equal. In [10] it was shown that using an Additive Increase Multiplicative Decrease (AIMD) adjustment policy results in convergence to a fair equilibrium. Thus, the algorithm proposed in this paper utilizes AIMD (described in below in Section III.B step 4) to ensure convergence to fairness. The load fairness is measured using the following fairness index defined in [10]:

$$\text{Fairness} = \frac{\left(\sum \mu_i\right)^2}{N \sum \mu_i^2}, \quad (4)$$

where N is the total number of equivalent nodes sharing the load and μ_i denotes the average wake-up rate of node i . A completely fair system will have fairness equal to one and a system where the load is equally distributed between m nodes will have a fairness of m/N . Note that a completely unfair system ($m = 1$) has fairness equal to $1/N$.

IV. ALGORITHM

The algorithm is executed independently at every node as soon as the node wakes up. For a node i located in a region B the algorithm consists of the following steps:

1. Wake up and announce that it ready to process packets.
2. Receive all k waiting packets, over a short, fixed period. Note that node i does not accept additional packets after this period.
3. Use k and t_0 , the longest time a packet has been waiting, to estimate the time $1/\hat{\mu}_B$ since the last wake-up in region B . ($\hat{\mu}_B$ is node i 's estimate of combined rate of nodes waking up in region B).
4. If $\hat{\mu}_B$ for the region is less than the optimal, μ^* , node i multiplicatively decreases its average sleep time, $1/\mu_i$. Conversely, if the estimated wake up rate in the region is greater than μ^* , node i additively increases its average sleep time.
5. Stay awake until all k packets have been forwarded (exhaustive service discipline).
6. Draw a value x from an exponential distribution with mean $1/\mu_i$ and go to sleep for the duration of x . When node i awakens return to step 1.

V. SIMULATIONS

A. Implementation details

The simulations were conducted using OmNet++[11], a discrete event simulator developed by Andras Varga at the Technical University of Budapest. Results are based on simulations of ad-hoc networks, having 700-950 nodes each. The network topology consists of nodes arranged in a 2-dimensional rectangular region with one sink node in the center of the region. During the simulation period, the nodes generate data packets that are routed to the sink node. Each simulation period lasts for 10,000 seconds. The node density and channel characteristics are varied to study their effects on the sleeping discipline.

The fading channels are modeled using a Bernoulli model with parameter p . In other words, each time two nodes attempt to communicate, the probability of the communication succeeding is p . The success parameter p is also referred to as the channel quality.

In order to simplify opportunistic routing, the network is divided into a grid of blocks. It is assumed that the nodes know their geographical location a priori and can determine which block they are located in. The nodes in a particular block are equivalent from a routing perspective. The routing in the simulation is done from block-to-block. That is, packets are routed from a source/forwarding node in block b , to any equivalent node in block c , provided that block c is geographically closer to the destination than block b and the two blocks are adjacent.

In the simulation, the probabilistic delay constraint (the bound on the delay of a single routing hop) is set to 0.2 seconds. In other words, the probability of a single-hop delay being greater than 0.2 seconds should be less than 1%. Substituting these values into (1) and solving for μ^* , give 23.3 wake-ups per second as the optimal node wake-up rate in a region. The channel quality affects the actual wake-up rate of the nodes. The lower the probability p of communicating with another node, the more often the nodes need to wake-up to

maintain the same effective wake-up rate. For a given p , the actual wake-up rate of the nodes should be $23.3/p$.

In order to determine if the actual wake-up rate matches the optimal rate μ^*/p , each node needs to form a reliable estimate of the aggregate node wake-up rates. In the simulation, exponential low-pass filtering ($R = \alpha R + M$, where R is the time-average wake-up rate estimation and M is a latest estimation of the wake-up rate) is used at each node.

In a fair network, all nodes share the load equally. Thus the optimal contribution for an individual node is $\mu_i^* = \mu^*/p * (\# \text{equivalent nodes}) = \mu^*/p * (\text{density} * \text{block area})$. This optimal contribution, μ_i^* , can be expressed as an optimal duty-cycle. This duty-cycle is given by the amount of time the node stays on divided by μ_i^* . In the simulation the amount of time the node stays on is the short fixed observation period plus the time taken to transmit all packets (expected per-hop packet delay). The optimal duty-cycle per node is included in the results to evaluate the performance of the algorithm.

B. Results

The results section demonstrates the following important aspects of the algorithm: operation within the delay constraint,

efficiency of operation relative to the optimal wake-up rate, fairness, robustness and adaptation to varying node densities and communication channels.

First, the results show that the algorithm does meet the desired delay constraint. Table 1 shows the average of the one-hop delays and the percentage of one-hop delays that were greater than 0.2 seconds for a range of network configurations. For every network configuration the percentage of delays greater 0.2 seconds is less than 1%. This falls within the 99% probabilistic delay bound. Figure 1 shows the distribution of the one-hop delays experienced by packets during the simulation. As expected, given the Poisson distribution of the node-wake-ups, the shape of the curve is exponential. An application is more likely to require an end-to-end delay bound than a per-hop delay bound. Thus, the relationship between the one-hop delay and the end-to-end delay is important. Figure 3 shows the average end-to-end delay as a function of the number of hops. The end-to-end delay is a linear function of the number of hops. Therefore, an application can use the one-hop delay bound provided by the algorithm, along with knowledge of the diameter of the network to create an end-to-end delay bound.

TABLE I. DUTY-CYCLE, AVERAGE ONE-HOP DELAY, PERCENTAGE DELAYS OVER 0.2S AND FAIRNESS AS A FUNCTION OF DENSITY AND CHANNEL QUALITY OF THE NETWORK.

Density	Channel quality	Observed Duty-cycle	Optimal Duty-Cycle	Avg delay	% delay > 0.2s	Fairness
10	0.4	21.06	17.9	0.030	0.25	0.99
10	0.6	14.81	12.1	0.029	0.31	0.98
10	0.8	11.02	9.2	0.029	0.47	0.93
15	0.4	16.11	12.1	0.020	0.18	0.98
15	0.6	10.73	8.2	0.026	0.29	0.93
15	0.8	8.02	6.2	0.026	0.47	0.86
20	0.4	11.11	9.2	0.029	0.47	0.93
20	0.6	7.52	6.2	0.03	0.7	0.85
20	0.8	7.0	4.8	0.023	0.27	0.92

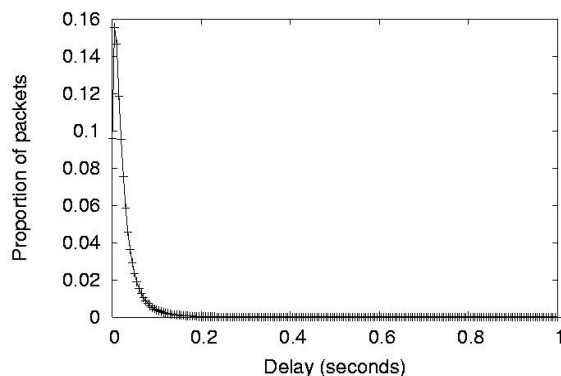


Figure 2. Distribution of Packet Delays.

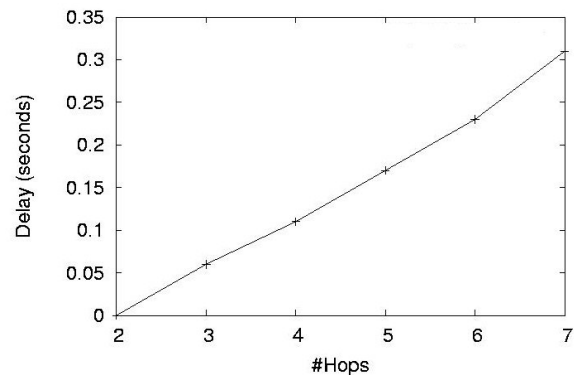


Figure 3. End-to-End delay as a function of number of hops.

Efficiency is another important aspect of the algorithm. Table 1 shows that the actual duty-cycle is about 3% larger than the optimal duty-cycle. This overhead can be attributed to the fact that the actual one-hop delay performance is less than the delay constraint. This slight overshooting of the delay-constraint can be expected due to oscillations of the (AIMD)

feedback-control loop implemented in the simulation. The high (0.85+) fairness indices in Table 1 also demonstrate that the AIMD in the algorithm does ensure convergence to a fair equilibrium.

The algorithm's robustness to changing topologies is evident from Table 1. For different values of the channel quality p , the algorithm continues to meet the delay constraint, but the resulting duty-cycles differ. As discussed in Section III, the duty-cycle is inversely proportional to the channel quality p . In addition, the duty-cycle is also inversely proportional to the node density d . In fact, for different p 's and d 's which have a constant product pd the duty-cycle should be the same. This is illustrated by comparing rows 2 & 4, 3 & 7, and 6 & 8 of Table 1. Specifically, examining rows 3 & 7, with $d_3=10$, $p_3=0.8$, $d_7=20$, $p_7=0.4$, reveals that the resulting duty-cycles, 11.02 and 11.11 respectively, are almost equal. Thus, channel quality is equivalent to density; lowering the channel quality has the same effect on the nodes' sleeping time as lowering the density of nodes.

Lastly, Figure 4 demonstrates the adaptation of the algorithm to changing node density and channel quality. Figure 4 shows the time-average duty-cycle of all the nodes in a block. Initially, the channel quality is 0.7 and there are 16 nodes in the block. The first change to the network occurs at $t=2500$ s when the node density of the block is halved to 8 nodes. At $t=5000$ s, the node density is increased to 12 nodes. The jumps of the duty-cycle demonstrate the real-time adaptation of the algorithm to node-density changes. The last change occurs at $t=7500$ s when the channel quality drops to 0.4. Again the algorithm allows the nodes to adaptively increase their duty-cycles to the new channel quality.

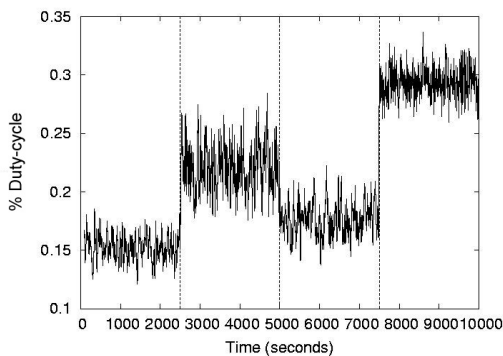


Figure 4. Average duty cycle as the network topology and channel quality changes.

VI. CONCLUSION

This paper presents an approach for exploiting high density of homogeneous nodes in a sensor network to allow nodes to conserve energy by maximizing their sleeping time. The algorithm developed in the paper ensures that application constraints, such as latency, can be met while nodes are

sleeping. The algorithm incorporates robustness to changing network connectivity and topology through randomized sleeping times, and makes sure that the load is evenly distributed among nodes. The overhead of the algorithm is extremely low and nodes are not required to keep state about individual neighbors or coordinate with their neighbors. Nodes independently determine how often to be awake based solely on the traffic patterns they observe while awake.

Simulations show that the algorithm allows nodes to sleep 90% of the time at moderate densities, while still satisfying the delay constraints of the traffic. The system is also robust to unreliable communication channels. The effect of unreliable channels is shown to be equivalent to operating at lower densities. Results also show that unreliable channels or low density merely forces the nodes to be awake more often. The algorithm fairness of the algorithm (i.e., balancing the load among equivalent nodes) is also demonstrated.

ACKNOWLEDGMENT

The authors would like to thank Prof. Adam Wolisz for his discussion and guidance. He has been essential in the development of this work.

REFERENCES

- [1] J. Rabaey et al., "PicoRadio supports ad hoc ultra-low power wireless networking," *IEEE Computer Magazine*, July 2000.
- [2] J. Kahn, R. Katz, and K. Pister, "Next century challenges: mobile networking for smart dust," *MobiCom*, 1999.
- [3] D. Estrin et al., "Embedded everywhere: a research agenda for networked systems of embedded computers," *Computer Science and Telecommunications Board (CSTB) Report*, 2001.
- [4] P. Ishwar, A. Kumar, and K. Ramchandran, "Distributed sampling for dense sensor networks: a 'bit-conservation principle'," *Proceedings of the Second International Workshop, IPSN 2003, Palo Alto, CA, USA, April 22-23*, pp. 17-31.
- [5] R. Stutz, "Performance analysis and optimization of a 2.4GHz, multi-hop, wireless self-configurable network", M.S. Thesis, Ecole Polytechnique Federale de Lausanne, 2003.
- [6] Y. Xu, J. Heidemann, and D. Estrin, "Geography-informed energy conservation for ad hoc routing," *Proc. MobiCom 2001*, pp. 70-84, July 2001.
- [7] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris, "Span: an energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks," *MobiCom 2001*.
- [8] F. Koushanfar, A. Davare, D. Nguyen, M. Potkonjak, A. Sangiovanni-Vincentelli, "Low power coordination in wireless ad-hoc networks" *International Symposium on Low Power Electronics and Design (ISLPED)*, August 2003
- [9] G. Grimmett and D. Stirzaker, *Probability and Random Processes*, Oxford Science Publications, 1992.
- [10] D.-M. Chiu and R. Jain, "Analysis of the increase and decrease algorithms for congestion avoidance in computer networks", *Computer Networks and ISDN Systems*, Vol. 17, 1989, pp. 1-14.
- [11] A. Varga, "The OMNeT++ discrete event simulation system," in *European Simulation Multiconference June 2001*.