Three Power-aware Routing Algorithms for Sensor Networks

Javed Aslam, Qun Li, Daniela Rus
Department of Computer Science
Dartmouth College
Hanover, NH 03755
{jaa, liqun, rus}@cs.dartmouth.edu

July 16, 2002

Abstract

This paper discusses online power-aware routing in large wireless ad-hoc networks (especially sensor networks) for applications where the message sequence is not known. We seek to optimize the lifetime of the network. We show that online power-aware routing does not have a constant competitive ratio to the off-line optimal algorithm. We develop an approximation algorithm called *max-min* zP_{min} that has a good empirical competitive ratio. To ensure scalability, we introduce a second online algorithm for power-aware routing. This hierarchical algorithm is called zone-based routing. Our experiments show that its performance is quite good. Finally, we describe a distributed version of this algorithm that does not depend on any centralization.

1 Introduction

The proliferation of low-power analog and digital electronics has created huge opportunities for the field of wireless computing. It is now possible to deploy hundreds of devices of low computation, communication and battery power. They can create ad-hoc networks and be used as distributed sensors to monitor large geographical areas, as communication enablers for field operations, or as grids of computation. These applications require great care in the utilization of power. The power level is provided by batteries and thus it is finite. Every message sent and every computation performed drains the battery.

In this paper we examine a class of algorithms for routing messages in wireless networks subject to power constraints and optimization. We envision a large ad-hoc network consisting of thousands of computers such as a sensor network distributed over a large geographical area. Clearly this type of network has a high degree of redundancy. We would like to develop a power-aware approach to routing messages in such a system that is fast, scalable, and is online in that it *does not know ahead of time the sequence of messages* that has to be routed over the network.

The power consumption of each node in an ad-hoc wireless system can be divided according to functionality into: (1) the power utilized for the transmission of a message; (2) the power utilized for the reception of a message; and (3) the power utilized while the system is idle. Table 1 lists power consumption numbers for several wireless cards. This suggests two complementary levels at which power consumption can be optimized: (1) minimizing power consumption during the idle time and (2) minimizing power consumption during communication. In this paper we focus only on issues related to minimizing power consumption during communication - that is, while the system is transmitting and receiving messages. We believe that efficient message routing algorithms, coupled with good solutions for optimizing power consumption during the idle time will lead to effective power management in wireless ad-hoc networks, especially for a sparsely deployed network.

Card	Tr	Rv	Idle	Slp	Power
	mA	mA	mA	mA	Sup. V
RangeLAN2-7410	265	130	n/a	2	5
WaveLAN(11Mbps)	284	190	156	10	4.74
Smart Spread	150	80	n/a	5	5

Table 1: Power Consumption Comparison among Different Wireless LAN Cards ([2, 12, 1]). For RangeLAN2, the power consumption for doze mode (which is claimed to be network aware) is 5mA. The last one is Smart Spread Spectrum of Adcon Telemetry.

Several metrics can be used to optimize power-routing for a sequence of messages. Minimizing the energy consumed for each message is an obvious solution that optimizes locally the power consumption. Other useful metrics include minimizing the variance in each computer power level, minimizing the ratio of cost/packet, and minimizing the maximum node cost. A drawback of these metrics is that they focus on individual nodes in the system instead of the system as a whole. Therefore, routing messages according to them might quickly lead to a system in which nodes have high residual power but the system is not connected because some critical nodes have been depleted of power. We choose to focus on a global metric by maximizing the lifetime of the network. We model this as the time to the earliest time a message cannot be sent. This metric is very useful for ad-hoc networks where each message is important and the networks are sparsely deployed.

In this paper we build on our previous work [22] and show that the online power-aware message routing problem is very hard (Section 3). This problem does not have a constant competitive ratio to the off-line optimal algorithm that knows the message sequence. Guided by this theoretical result, we propose an online approximation algorithm for power-aware message routing that optimizes the lifetime of the network and examine its bounds (Section 4). Our algorithm, called the *max-min* zP_{min} algorithm, combines the benefits of selecting the path with the minimum power consumption and the path that maximizes the minimal residual power in the nodes of the network. Despite the discouraging theoretical result concerning the competitive ratio for online routing, we show that the *max-min* zP_{min} algorithm has a good competitive ratio in practice, approaching the performance of the optimal off-line routing algorithm under realistic conditions.

Our proposed *max-min* zP_{min} algorithm requires information about the power level of each computer in the network. Knowing this information accurately is not a problem in small networks. However, for large networks it is difficult to aggregate and maintain this information. This makes it hard to implement the *max-min* zP_{min} algorithm for large networks. Instead, we propose another online algorithm called *zone-based routing* that relies on *max-min* zP_{min} and is scalable (Section 5). Our experiments show that the performance of zone-base routing is very close to the performance of *max-min* zP_{min} with respect to optimizing the lifetime of the network.

Zone-base routing is a hierarchical approach where the area covered by the (sensor) network is divided into a small number of zones. Each zone has many nodes and thus a lot of redundancy in routing a message through it. To send a message across the entire area we find a "global" path from zone to zone and give each zone control over how to route the message within itself. Thus, zone-based power-aware routing consists of (1) an algorithm for estimating the power level of each zone; (2) an algorithm computing a path for each message across zones; and (3) an algorithm for computing the best path for the message within each zone (with respect to the power lifetime of the zone.)

The algorithm *max-min* zP_{min} has the great advantage of not relying on the message sequence but the disadvantage of being centralized and requiring knowledge of the power level of each node in the system. These are unrealistic assumptions for field applications, for example involving sensor networks, where the computation is distributed and information localized. The third type of routing we describe is a distributed version of our centralized algorithms. distributed version of the *max-min* zP_{min} algorithm has the flavor of the distributed Bellman-Ford algorithm. This distributed algorithm requires n message broadcasts for each node if there is no clock synchronization, and only one message broadcast if the host clocks are synchronized.

2 Related Work

We are inspired by exciting recent results in ad-hoc networks and in sensor networks. Most previous research on ad-hoc network routing [19, 15, 24, 25, 27, 31, 20] focused on the protocol design and performance evaluation in terms of the message overhead and loss rate. To improve the scalability of routing algorithms for large networks, many hierarchical routing methods have been proposed in [21, 10, 23, 4, 13, 29, 36]. In [26, 18], zones, which are the route maintenance units, are used to find the routes. This previous work focused on how to find the correct route efficiently, but did not consider optimizing power while sending messages.

Singh et al. [32] proposed power-aware routing and discussed different metrics in power-aware routing. Some of the ideas in this paper are extensions of what that paper proposed. Minimal energy consumption was used in [30]. Stojmenovic and Lin proposed the first localized power-aware algorithm in their paper series [33]. Their algorithm is novel in combining the power and cost into one metric and running only based on the local information. Chang and Tassiulas [5] also used the combined metric to direct the routing. Their algorithm is proposed to maximize the lifetime of a network when the message rate is known. Their main idea, namely to avoid using low power nodes and choose the short path at the beginning, has inspired the approach described in this paper. We also use the same formula to describe the residual power fraction. The work presented in this paper is different from these previous results in that we develop online, hierarchical, and scalable algorithms that do not rely on knowing the message rate and optimize the lifetime of the network. In [14], Gupta and Kumar discussed the critical power at which a node needs to transmit in order to ensure the network is connected. Energy efficient MAC layer protocols can be found in [9, 8, 39]. Wu et al.[35] proposed the power-aware approach in dominating set based routing. Their idea is to use rules based on energy level to prolong the lifetime of a node in the refining process of reducing the the number of nodes in the dominating set.

Another branch of the related work concerns optimizing power consumption during idle time rather than during the time of communicating messages [38, 6].These protocols put some nodes in the network into sleep mode to conserve energy, while maintaining the connectivity of the network to ensure communication. In a related work [35, 37], Wu and Stojmenovic give an elegant solution by using connecting dominating sets, which generalize the idea of maintaining a connected network while keeping most of the nodes in sleeping mode. This work is complementary to the results of the idle time power conservation optimizing methods. Combined, efficient ways for dealing with idle time and with communication can lead to powerful power management solutions.

Work on reducing the communication overhead in broadcasting tasks [34] bears similarity with our approach to reducing the message broadcasting in routing application. In Stojmenovic et al.'s paper, a node will rebroadcast a message only if there are neighbors who are not covered by the previous broadcasts. In contrast, our distributed algorithms eliminate the message broadcasts that are useless by discerning them with the message delay. As a result, in some algorithms we proposed, we can get a constant message broadcasts for each node.

Related results in sensor networks include [28, 3, 17, 11, 16, 7]. The high-level vision of wireless sensor networks was introduced in [28, 3]. Achieving energy-efficient communication is an important issue in sensor network design. Using directed diffusion for sensor coordination is described in [17, 11]. In [16] a low-energy adaptive protocol that uses data fusion is proposed for sensor networks. Our approach is different from the previous work in that we consider message routing in sensor networks and our solution does not require to know or aggregate the data transmitted.

3 Formulation of Power-aware Routing

3.1 The Model

Power consumption in ad-hoc networks can be divided into two parts: (1) the idle mode and (2) the transmit/receive mode. The nodes in the network are either in idle mode or in transmit/receive mode at all time. The idle mode corresponds to a baseline power consumption. Optimizing this mode is the focus of [38, 6, 35, 37]. We instead focus on studying and optimizing the transmit/receive mode. When a message is routed through the system, all the nodes with the exception of the source and destination receives a message and then immediately relay it. Because of this, we can view the power consumption at each node as an aggregate between transit and receive powers which we will model as one parameter.

More specifically, we assume an ad-hoc network that can be represented by a weighted graph $G(V, E)$. The vertices of the graph correspond to computers in the network. They have weights that correspond to the computer's power level. The edges in the graph correspond to pairs of computers that are in communication range. Each edge weight is the power cost of sending a unit message[1] between the two nodes. Our results are independent of the power consumption model as long as we assume the power consumption of sending a unit message between two nodes does not change during a run of the algorithm. That is, the weight of any edge in the network graph is fixed.

[1]Without loss of generality, we assume that all the messages are unit messages. Longer messages can be expressed as sequences of unit messages.

Although our algorithms are independent of the power consumption model, we fixed one model for our implementation and simulation experiments. Suppose a host needs power e to transmit a message to another host who is d distance away. We use the model of [12, 16, 30] to compute the power consumption for sending this message:

$$e = kd^c + a,$$

where k and c are constants for the specific wireless system (usually $2 \leq c \leq 4$), and a is the electronics energy that depends on factors such as digital coding, modulation, filtering, and spreading of the signal. Since our algorithms can use any power consumption model, we use $a = 0$ to simplify the implementation.

We focus on networks where power is a finite resource. Only a finite number of messages can be transmitted between any two hosts. We wish to solve the problem of routing messages so as to maximize the battery lives of the hosts in the system. The lifetime of a network with respect to a sequence of messages is the earliest time when a message cannot be sent due to saturated nodes. We selected this metric under the assumption that all messages are important. Our results, however, can be relaxed to accommodate up to m message delivery failures, with m a constant parameter.

3.2 Relationship to Classical Network Flow

Power-aware routing is different from the maximal network flow problem although there are similarities. The classical network flow problem constrains the capacity of the edges instead of limiting the capacity of the nodes. If the capacity of a node does not depend on the distances to neighboring nodes, our problem can also be reduced to maximal network flow.

We use the following special case of our problem in which there is only one source node and one sink node to show the problem is NP-hard. The maximal number of messages sustained by a network from the source nodes to the sink nodes can be formulated as linear programming. Let n_{ij} be the total number of messages from node v_i to node v_j, e_{ij} denote the power cost to send a message between node v_i to node v_j, and s and t denote the source and sink in the network. Let P_i denote the power of node i. We wish to maximize the number of messages in the system subject to the following constraints: (1) the total power used to send all messages from node v_i does not exceed P_i; and (2) the number of messages from v_i to all other nodes is the same as the number of messages from all other nodes to v_i, which are given below:

$$\begin{aligned} & \textit{maximize} \qquad \sum_j n_{sj} \;\; \textit{subject to} \\ \sum_j n_{ij} \cdot e_{ij} \;&\leq\; P_i \qquad (1) \\ \sum_j n_{ij} \;&=\; \sum_j n_{ji} \;\; (\textit{for } i \neq s,\ t) \qquad (2) \end{aligned}$$

This linear programming formulation can be can be solved in polynomial time. However, we need the integer solution, but computing the integer solution is NP-hard. Figure 1 shows the reduction to set partition for proving the NP-hardness of the integer solution.

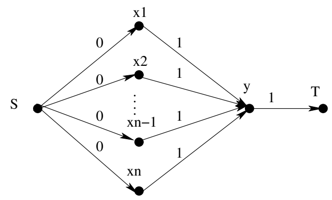

Figure 1: The integer solution problem can be reduced to set partition as follows. Construct a network based on the given set. The power of x_i is a_i for all $1 \leq i \leq n$, and the power of y is $\sum_{a_i \in A} a_i/2$. The weight of each edge is marked on the network. For any set of integers $S = a_1, a_2, \cdots, a_n$, we are asked to find the subset of S, A such that $\sum_{a_i \in A} a_i = \sum_{a_i \in S-A} a_i$. We can construct a network as depicted here. The maximal flow of the network is $\sum_{a_i \in A} a_i/2$, and it can only be gotten when the flow of $x_i y$ is a_i for all $a_i \in A$, and for all other $x_i y$, the flow is 0.

3.3 Competitive Ratio for Online Power-aware Routing

In a system where the message rates are unknown, we wish to compute the best path to route a message. Since the message sequence is unknown, there is no guarantee that we can find the optimal path. For example, the path with the least power consumption can quickly saturate some of the nodes. The difficulty of solving this problem without knowledge of the message sequence is summarized by the theoretical properties of its competitive ratio. The competitive ratio of an online algorithm is the ratio between the performance of that algorithm and the optimal off-line algorithm that has access to the entire execution sequence prior to making any decisions.

Theorem 1 *No online algorithm for message routing has a constant competitive ratio in terms of the lifetime of the network or the number of messages sent.*

Theorem 1, whose proof is shown in Figure 2, shows that it is not possible to compute online an optimal solution for power-aware routing.

4 Online Power-aware Routing with $max\text{-}min\ \ zP_{min}$

In this section we develop an approximation algorithm for online power-aware routing and show experimentally that our algorithm has a good empirical competitive ratio and comes close to the optimal.

We believe that it is important to develop algorithms for message routing that do not assume prior knowledge of the message sequence because for ad-hoc network applications this sequence is dynamic and depends on sensed values and goals communicated to the system as needed. Our goal is to increase the lifetime of the network when the message sequence is not known. We model lifetime as the earliest time that a message cannot be sent. Our assumption is that each message is important and thus the failure of delivering a message is a critical event. Our results can be extended to tolerate up to m message delivery failures, where m is a parameter. We focus the remaining of this discussion on the failure of the first message delivery.

Intuitively, message routes should avoid nodes whose power is low because overuse of those nodes will deplete their battery power. Thus, we would like to route messages along the path

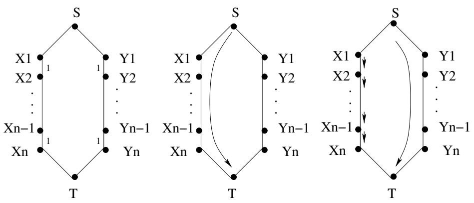

Figure 2: In this network, the power of each node is $1+\epsilon$ and the weight on each edge is 1. The first figure gives the network; the center one is the route for the online algorithm; and the right one is the route for the optimal algorithm. Consider the message sequence that begins with a message from S to T, say, ST. Without loss of generality (since there are only two possible paths from S to T), the online algorithm routes the message via the route $SX_1X_2X_3\cdots X_{n-1}X_nT$. The message sequence is X_1X_2, X_2X_3, X_3X_4, $\cdots$, $X_{n-1}X_n$. It is easy to see that the optimal algorithm (see right figure) routes the first message through $SY_1Y_2Y_3\cdots Y_{n-1}Y_nT$, then routes the remaining messages through X_1X_2, X_2X_3, X_3X_4, $\cdots$, and $X_{n-1}X_n$. Thus the optimal algorithm can transmit n messages. The online algorithm (center) can transmit at most 1 message for this message sequence because the nodes $X_1, X_2, \cdots, X_n$ are all saturated after routing the first message. The competitive ratio is small when n is large.

with the maximal minimal fraction of remaining power after the message is transmitted. We call this path the *max-min path*. The performance of max-min path can be very bad, as shown by the example in Figure 3. Another concern with the max-min path is that going through the nodes with high residual power may be expensive as compared to the path with the minimal power consumption. Too much power consumption decreases the overall power level of the system and thus decreases the life time of the network. There is a trade-off between minimizing the total power consumption and maximizing the minimal residual power of the network. We propose to enhance a max-min path by limiting its total power consumption.

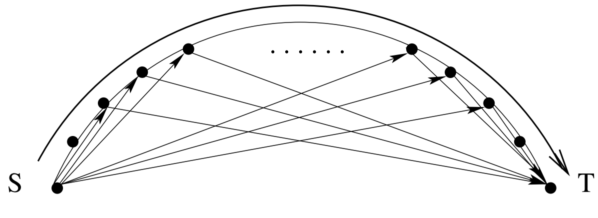

Figure 3: The performance of max-min path can be very bad. In this example, each node except for the source S has the power $20+\epsilon$, and the weight of each edge on the arc is 1. The weight of each straight edge is 2. Let the power of the source be ∞. The network can send 20 messages from S to T according to max-min strategy by taking the edges on the arc (see the arc on the top). But the optimal number of messages follows the straight edges with black arrows is $10(n-4)$ where n is the number of nodes.

The two extreme solutions to power-aware routing for one message are: (1) compute a path with minimal power consumption P_{min}; and (2) compute a path that maximizes the minimal residual power in the network. We look for an algorithm that optimizes *both* criteria. We relax the minimal power consumption for the message to be zP_{min} with parameter $z \geq 1$ to restrict

Algorithm 1 *max-min* zP_{min}-path algorithm

Find the path with the least power consumption, P_{min} by using the Dijkstra algorithm
while true **do**
  Find the path with the least power consumption in the graph
  if the power consumption $> z \cdot P_{min}$ or no path is found **then**
    the previous shortest path is the solution, stop
  Find the minimal u_{tij} on that path, let it be u_{min}
  Find all the edges whose residual power fraction $u_{tij} \leq u_{min}$, remove them from the graph

the power consumption for sending one message to zP_{min}. We propose an algorithm we call *max-min* zP_{min} that consumes at most zP_{min} while maximizing the minimal residual power fraction. The rest of the section describes the *max-min* zP_{min} algorithm, presents empirical justification for it, a method for adaptively choosing the parameter z and describes some of its theoretical properties.

The following notation is used in the description of the *max-min* zP_{min} algorithm. Given a network graph (V, E), let $P(v_i)$ be the initial power level of node v_i, e_{ij} the weight of the edge v_iv_j, and $P_t(v_i)$ is the power of the node v_i at time t. Let $u_{tij} = \frac{P_t(v_i)-e_{ij}}{P(v_i)}$ be the residual power fraction after sending a message from i to j.

Alg. 1 describes the algorithm. In each round we remove at least one edge from the graph. The algorithm runs the Dijkstra algorithm to find the shortest path for at most $|E|$ times where $|E|$ is the number of edges. The running time of the Dijkstra algorithm is $O(|E| + |V| \log |V|)$ where $|V|$ is the number of nodes. Then the running time of the algorithm is at most $O(|E| \cdot (|E| + |V| \log |V|))$. By using binary search, the running time can be reduced to $O(\log |E| \cdot (|E| + |V| \log |V|))$. To find the pure max-min path, we can modify the Bellman-ford algorithm by changing the relaxation procedure. The running time is $O(|V| \cdot |E|)$.

4.1 Adaptive Computation for z

An important factor in the *max-min* zP_{min} algorithm is the parameter z which measures the trade-off between the max-min path and the minimal power path. When $z = 1$ the algorithm computes the minimal power consumption path. When $z = \infty$ it computes the max-min path. We would like to investigate an adaptive way of computing $z > 1$ such that *max-min* zP_{min} that will lead to a longer lifetime for the network than each of the max-min and minimal power algorithms. Alg. 2 describes the algorithm for adaptively computing z. P is the initial power of a host. ΔP_t is the residual power decrease at time t compared to time $t - T$. Basically, $\frac{P}{\Delta P_t}$ gives an estimation for the lifetime of that node if the message sequence is regular with some cyclicity. The adaptive algorithm works well when the message distributions are similar as the time elapses.

We conducted several simulation experiments to evaluate the adaptive computation of z. In a first experiment we generated the positions of hosts in a square field randomly using the following parameters. The scope of the network is $10 * 10$, the number of hosts in the network is 20, the power consumption weights for transmitting a message are $e_{ij} = 0.001 * d_{ij}^3$, and the initial power of each host is 30. Messages are generated between all possible pairs of hosts and are distributed evenly. Figure 4 (first) shows the number of messages transmitted until the first message delivery failure for different values of z. Using the adaptive method for selecting z with $z_{init} = 10$, the total number of messages sent increases to $12,207$, which is almost the

Algorithm 2 Adaptive *max-min* zP_{min} algorithm

Choose initial value z, the step δ
Run the *max-min* zP_{min} algorithm for some interval T
Compute $\frac{P}{\Delta P_t}$ for every host, let the minimal one be t_1
while true **do**
  Increase z by δ, and run the algorithm again for time T
  Compute the minimal $\frac{P}{\Delta P_t}$ among all hosts, let it be t_2
  if some host is saturated **then**
    exit
  if $t_1 < t_2$ **then**
    $t_1 = t_2$
  if $t_1 > t_2$ **then**
    $\delta = -\delta/2$, $t_1 = t_2$

best performance by *max-min* zP_{min} algorithm.

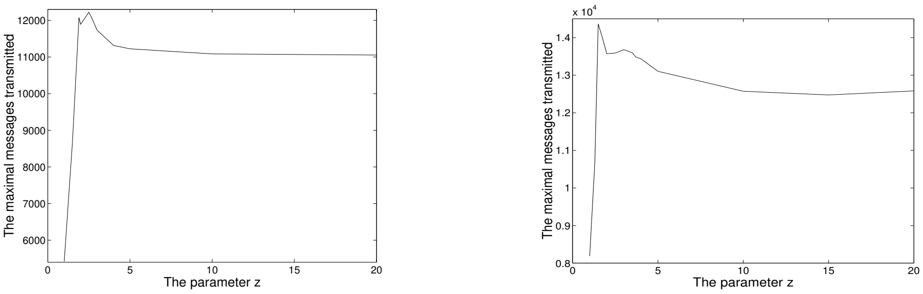

Figure 4: The effect of z on the maximal number of messages in a square network space. The positions of hosts are generated randomly. In the first graph the network scope is $10 * 10$, the number of hosts is 20, the weights are generated by $e_{ij} = 0.001 * d_{ij}^3$, the initial power of each host is 30, and messages are generated between all possible pairs of the hosts and are distributed evenly. In the second graph the number of hosts is 40, the initial power of each node is 10, and all other parameters are the same as the first graph.

In the second experiment we generated the positions of hosts evenly distributed on the perimeter of a circle. The radius of the circle is 20, number of hosts 20; the weight formula: $e_{ij} = 0.0001 * d_{ij}^3$; and the initial power of each host is 10. Messages are generated between all possible pairs of the hosts and are distributed evenly. The performance according to various z can be found in Figure 5 (first). By using the adaptive method, the total number of messages sent until reaching a network partition is $11,588$, which is much better than the most cases when we choose a fixed z.

4.2 Empirical Evaluation of Max-min zP_{min} Algorithm

We conducted several experiments for evaluating the performance of the *max-min* zP_{min} algorithm.

In the first set of experiments (Figure 4), we compare how z affects the performance of the lifetime of the network. In the first experiment, a set of hosts are randomly generated on a

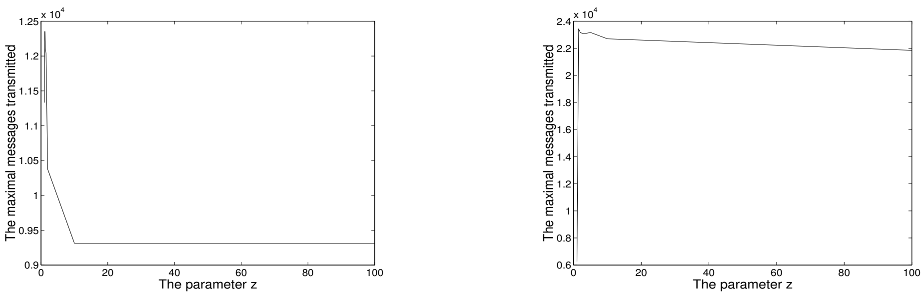

Figure 5: The first figure shows the effect of z on the maximal number of messages in a ring network. The radius of the circle is 20, the number of hosts is 20, the weights are generated by $e_{ij} = 0.0001 * d_{ij}^3$, the initial power of each host is 10 and messages are generated between all possible pairs of the hosts and are distributed evenly. The second figure shows a network with four columns of the size $1 * 0.1$. Each area has ten hosts which are randomly distributed. The distance between two adjacent columns is 1. The right figure gives the performance when z changes. The vertical axis is the maximal messages sent before the first host is saturated. The number of hosts is 40; the weight formula is $e_{ij} = 0.001 * d_{ij}^3$; the initial power of each host is 1; messages are generated between all possible pairs of the hosts and are distributed evenly.

square. For each pair of nodes, one message is sent in both directions for a unit of time. Thus there is a total of $n * (n - 1)$ messages sent in each unit time, where n is the number of the hosts in the network. We experimented with other network topologies. Figure 5 (first) shows the results obtained in a ring network. Figure 5 (second) shows the results obtained when the network consists of four columns where nodes are approximately aligned in each column. The same method used in experiment 1 varies the value of z.

These experiments show that adaptively selecting z leads to superior performance over the minimal power algorithm ($z = 1$) and the max-min algorithm ($z = \infty$). Furthermore, when compared to an optimal routing algorithm, *max-min* zP_{min} has a constant empirical competitive ratio (see Figure 6 (first)).

Figure 6 (second) shows more data that compares the *max-min* zP_{min} algorithm to the optimal routing strategy. We computed the optimal strategy by using a linear programming package[2]. We ran 500 experiments. In each experiment a network with 20 nodes was generated randomly in a $10 * 10$ network space. The messages were sent to one gateway node repeatedly. We computed the ratio of the lifetime of the *max-min* zP_{min} algorithm to the optimal lifetime. Figure 6 shows that $max - min \ \ zP_{min}$ performs better than 80% of optimal for 92% of the experiments and performs within more than 90% of the optimal for 53% of the experiments. Since the optimal algorithm has the advantage of knowing the message sequence, we believe that *max-min* zP_{min} is practical for applications where there is no knowledge of the message sequence.

[2]To compute the optimal lifetime, the message rates are known. The max-min algorithm does not have this information.

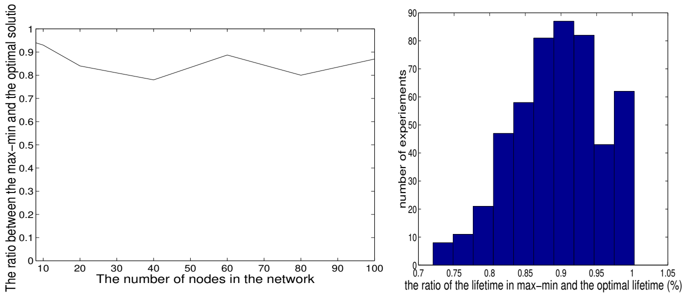

Figure 6: The first graph compares the performance of *max-min* zP_{min} to the optimal solution. The positions of hosts in the network are generated randomly. The network scope is $10 * 10$, the weight formula is $e_{ij} = 0.0001 * d_{ij}^3$, the initial power of each host is 10, messages are generated from each host to a specific gateway host, the ratio z is 100.0. The second figure shows the histogram that compares *max-min* zP_{min} to optimal for 500 experiments. In each experiment the network consists of 20 nodes randomly placed in a 10*10 network space. The cost of messages is given by $e_{ij} = 0.001 * d_{ij}^3$. The hosts have the same initial power and messages are generated for hosts to one gateway host. The horizontal axis is the ratio between the lifetime of the *max-min* zP_{min}max-min algorithm and the optimal lifetime, which is computed off-line.

4.3 Analysis of the *Max-min* zP_{min} Algorithm

In this section we quantify the experimental results from the previous section in an attempt to formulate more precisely our original intuition about the trade-off between the minimal power routing and max-min power routing. We provide a lower bound for the lifetime of the *max-min* zP_{min} algorithm as compared to the optimal solution. We discuss this bound for a general case where there is some cyclicity to the messages that flow in the system and then show the specialization to the no cyclicity case.

Suppose the message distribution is regular, that is, in any period of time $[t_1, t_1 + \delta)$, the message distributions on the nodes in the network are the same. Since in sensor networks we expect some sort of cyclicity for message transmission, we assume that we can schedule the message transmission with the same policy in each time slice we call δ. In other words, we partition the time line into many time slots $[0, \delta), [\delta, 2\delta), [2\delta, 3\delta), \cdots$. Note that δ is the lifetime of the network if there is no cyclical behavior in message transmission. We assume the same messages are generated in each δ slot but their sequence may be different.

Let the optimal algorithm be denoted by O, and the *max-min* zP_{min} algorithm be denoted by M. In M, each message is transmitted along a path whose overall power consumption is less than z times the minimal power consumption for that message. The initial time is 0. The lifetime of the network by algorithm O is T_O, and the lifetime by algorithm M is T_M. The initial power of each node is: P_{10}, P_{20}, P_{30}, $\cdots$, $P_{(n-1)0}$, P_{n0}. The remaining power of each node at T_O by running algorithm O is: P_{1O}, P_{2O}, P_{3O}, $\cdots$, P_{n-1O}, P_{nO}. The remaining power of each node at T_M by running algorithm M is: P_{1M}, P_{2M}, P_{3M}, $\cdots$, P_{n-1M}, P_{nM}. Let the message sequence in any slot be $m_1, m_2, \cdots, m_s$, and the minimal power consumption to transmit those messages be P_{0m_1}, P_{0m_2}, P_{0m_3}, $\cdots$, P_{0m_s}.

Theorem 2 *The lifetime of algorithm M satisfies*

$$T_M \geq \frac{T_O}{z} + \frac{\delta \cdot (\sum_{k=1}^{n} P_{kO} - \sum_{k=1}^{n} P_{kM})}{z \cdot \sum_{k=1}^{s} P_{0m_k}} \tag{3}$$

Proof: We have

$$\sum_{k=1}^{n} P_{k0} = \sum_{k=1}^{n} P_{kM} + \sum_{k=1}^{M_{T_M}} P_{Mm_k} = P_M$$

where M_{T_M} is the number of messages transmitted from time point 0 to T_M. P_{Mm_k} is the power consumption of the k-th message by running algorithm M. We also have:

$$\sum_{k=1}^{n} P_{k0} = \sum_{k=1}^{n} P_{kO} + \sum_{k=1}^{M_{T_O}} P_{Om_k} = P_O$$

where M_{T_O} is the number of messages transmitted from time point 0 to T_O. P_{Om_k} is the power consumption of of the k-th message by running algorithm O.

Since the messages are the same for any two slots without considering their sequence, we can schedule the messages such that the message rates along the same route are the same in the two slots (think about divide every message into many tiny packets, and average the message rate along a route in algorithm O into the two consecutive slots evenly.). We have:

$$\sum_{k=1}^{M_{T_O}} P_{Om_k} = \frac{M_{T_O}}{s} \cdot \sum_{k=1}^{s} P_{Om_k} = \frac{T_O}{\delta} \cdot \sum_{k=1}^{s} P_{Om_k}$$

and

$$\sum_{k=1}^{M_{T_M}} P_{Mm_k} = \sum_{j=1}^{T_M/\delta} \sum_{k=1}^{s} P_{Mmkj}$$

So we have:

$$P_O = \sum_{k=1}^{n} P_{kO} + \frac{T_O}{\delta} \cdot \sum_{k=1}^{s} P_{Om_k}$$

,

$$P_M = \sum_{k=1}^{n} P_{kM} + \sum_{j=1}^{T_M/\delta} \sum_{k=1}^{s} P_{Mmkj}$$

and

$$P_O = P_M$$

P_{Mmkj} is the power consumption of the k-th message in slot j by running algorithm M. We also have the following assumption and the minimal power of P_{0mk}. For any $1 \leq j \leq \frac{T_M}{\delta}$ and k, we have only one corresponding l,

$$P_{Mmkj} \leq z \cdot P_{0m_l} \text{ and } P_{Om_k} \geq P_{0m_k}$$

Then,

$$P_O \geq \sum_{k=1}^{n} P_{kO} + \frac{T_O}{\delta} \cdot \sum_{k=1}^{s} P_{0m_k}$$

$$P_M \leq \sum_{k=1}^{n} P_{kM} + \frac{z \cdot T_M}{\delta} \cdot \sum_{k=1}^{s} P_{0m_k}$$

Thus,

$$\sum_{k=1}^{n} P_{kM} + \frac{z \cdot T_M}{\delta} \cdot \sum_{k=1}^{s} P_{0m_k} \geq \sum_{k=1}^{n} P_{kO} + \frac{T_O}{\delta} \cdot \sum_{k=1}^{s} P_{0m_k}$$

We have:

$$T_M \geq \frac{T_O}{z} + \frac{\delta \cdot (\sum_{k=1}^{n} P_{kO} - \sum_{k=1}^{n} P_{kM})}{z \cdot \sum_{k=1}^{s} P_{0m_k}}$$

□

Theorem 2 gives us insight into how well the message routing algorithm does with respect to optimizing the lifetime of the network. Given a network topology and a message distribution, T_O, δ, $\sum_{k=1}^{n} P_{kO}$, $\sum_{k=1}^{s} P_{0m_k}$ are all fixed in Equation 3. The variables that determine the actual lifetime are $\sum_{k=1}^{n} P_{kM}$ and z. The smaller $\sum_{k=1}^{n} P_{kM}$[3] is, the better the performance lower bound is. And the smaller z is, the better the performance lower bound is. However, a small z will lead to a large $\sum_{k=1}^{n} P_{kM}$. This explains the trade-off between minimal power path and max-min path.

Theorem 2 can be used in applications that have a regular message distribution without the restriction that all the messages are the same in two different slots. For these applications, the ratio between δ and $\sum_{k=1}^{s} P_{0m_k}$ must be changed to $1/\sum_{k=1}^{r} P_{0m_k}$, where P_{0m_k} is the minimal power consumption for the message generated in a unit of time.

Theorem 3 *The optimal lifetime of the network is at most* $\frac{t_{SPT} \cdot \sum P_h}{\sum P_h - \sum P_h^{SPT}}$ *where* t_{SPT} *and* P_h^{SPT} *are the life time of the network and remaining power of host* h *by using the least power consumption routing strategy.* P_h *is the initial power of host* h*.*

Proof: $t_{OPT} \leq \frac{\sum P_h}{\sum P_m^{SPT}} = \sum P_h / (\frac{\sum P_h - \sum P_h^{SPT}}{t_{SPT}})$
$= \frac{t_{SPT} \cdot \sum P_h}{\sum P_h - \sum P_h^{SPT}}$ □

5 Hierarchical Routing using Zone-based *max-min* zP_{min}

Although it has very nice theoretical and empirical properties, *max-min* zP_{min} algorithm is hard to implement on large scale networks. The main obstacle is that *max-min* zP_{min} requires accurate power level information for all the nodes in the network. It is difficult to collect this information from all the nodes in the network. One way to do it is by broadcast, but this would generate a huge power consumption which defeats our original goals. Furthermore, it is not clear how often such a broadcast would be necessary to keep the network data current. In this section we propose a hierarchical approach to power-aware routing that does not use as much information, does not know the message sequence, and relies in a feasible way on *max-min* zP_{min}.

We propose to organize the network structurally in geographical zones, and hierarchically to control routing across the zones. The idea is to group together all the nodes that are in geographic proximity as a zone, treat the zone as an entity in the network, and allow each

[3]This is the remaining power of the network at the limit of the network.

zone to decide how to route a message across[4]. The hosts in a zone autonomously direct local routing and participate in estimating the zone power level. Each message is routed across the zones using information about the zone power estimates. In our vision, a global controller for message routing manages the zones. This may be the node with the highest power, although other schemes such as round robin may also be employed.

If the network can be divided into a relatively small number of zones, the scale for the global routing algorithm is reduced. The global information required to send each message across is summarized by the power level estimate of each zone. We believe that in sensor networks this value will not need frequent updates because observable changes will occur only after long periods of time.

The rest of this section discusses (1) how the hosts in a zone collaborate to estimate the power of the zone; (2) how a message is routed within a zone; and (3) how a message is routed across zones. (1) and (3) will use our *max-min* zP_{min} algorithm, which can be implemented in a distributed way by slightly modifying our definition of the *max-min* zP_{min} path. The $max - min$ algorithm used in (2) is basically the Bellman-Ford algorithm, which can also be implemented as a distributed algorithm.

5.1 Zone Power Estimation

The power estimate for each zone is controlled by a node in the zone. This estimation measures the number of messages that can flow through the zone. Since the messages come from one neighboring zone and get directed to a different neighboring zone, we propose a method in which the power estimation is done *relative to the direction* of message transmission.

The protocol employed by the controller node consists of polling each node for its power level followed by running the *max-min* zP_{min} algorithm. The returned value is then broadcast to all the zones in the system. The frequency of this procedure is inversely proportional to the estimated power level. When the power level is high, the power estimation update can be done infrequently because messages routed through the zone in this period will not change the overall power distribution in the entire network much. When the power level is low, message transmission through the zone is likely to change the power distribution significantly.

Without loss of generality, we assume that zones are square so that they have four neighbors pointed to the North, South, East, and West[5]. We assume further that it is possible to communicate between the nodes that are close to the border between two zones, so that in effect the border nodes are part of both zones. In other words, neighboring zones that can communicate with each other have an area of overlap (see Figure 7 (first)).

The power estimate of a zone can be approximated as follows. We can use the *max-min* zP_{min} algorithm to evaluate the power level, find the *max-min* zP_{min} path, simulate sending Δ messages through the path, and repeat until the network is saturated. Δ is chosen to be proportionate to the power level of the zone.

More precisely, consider Figure 7 (first). To estimate the power of zone B with respect to sending messages in the direction from A to C, let the left part of the overlap between A and B be the source area and the right part of the overlap between B and C the sink area. The power of zone B in the direction from A to C is the maximal number of messages that can flow from the source nodes to the sink nodes before a node in B gets saturated. This can be computed

[4]This geographical partitioning can be implemented easily using GPS information from each host.

[5]this method can easily be generalized to zones with finite number of neighboring zones.

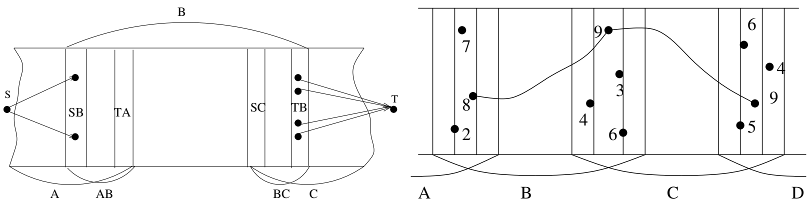

Figure 7: Three zones, A, B, and C. SB, SC are the source areas of B and C, and TA, TB are the sink areas of A and B. AB and BC are overlap border areas. The right figure shows how to connect the local path in zone B with the local path in zone C. The number next to each node is the number of paths passing through that node in the power evaluation procedure. The vertical stripes are the source and sink areas of the zones.

Algorithm 3 An approximation algorithm for zone power evaluation.

```
choose Δ for the message granularity. P = 0
while no node is depleted of power do
    Find the max-min zP_min path for Δ messages
    send the Δ messages through the zone
    P = P + Δ
return P
```

with the *max-min* zP_{min} algorithm (see Alg. 3). We start with the power graph of zone B and augment it. We create an imaginary source node S and connect it to all the source nodes. We create an imaginary sink node T and connect all the sink nodes to it. Let the weights of the newly added edges be 0. The *max-min* zP_{min} algorithm run on this graph determines the power estimate for zone B in the direction of A to C.

5.2 Global Path Selection

Given power-levels for each possible direction of message transmission, it is possible to construct a small zone-graph that models the global message routing problem. Figure 8 shows an example of a zone graph. A zone with k neighbors is represented by $k + 1$ vertices in this graph[6]. One vertex labels the zone; k vertices correspond to each message direction through the zone. The zone label vertex is connected to all the message direction vertices by edges in both direction. In addition, the message direction vertices are connected to the neighboring zone vertices if the current zone can go to the next neighboring zone in that direction. Each zone vertex has a power level of ∞. Each zone direction vertex is labeled by its estimated power level computed with the procedure in Section 5.1. Unlike in the model we proposed in Section 3.3, the edges in this zone graph do not have weights. Thus, the global route for sending a message can be found as the max-min path in the zone graph that starts in the originator's zone vertex and ends in the destination zone vertex for the message. We would like to bias towards path selection that uses the zones with higher power level. We can modify the Bellman-Ford algorithm (Alg. 4) to accomplish this.

[6]For square zones $k = 4 + 1$ as shown in Figure 8.

Algorithm 4 Maximal minimum power level path

Given graph $G(V, E)$, annotated with power level $p(v)$ for each $v \in V$
Find the path from s to t, $s = v_0, v_1, \cdots, v_{k-1}, v_k = t$ such that $min_{i=1}^{k-1} p(v_i)$ is maximal
for each vertex $v \in V[G]$ **do**
    if edge $(s, v) \in E[G]$ **then**
        $d[v] \leftarrow \infty$, $\pi[v] \leftarrow s$
    else
        $d[v] \leftarrow 0$, $\pi[v] \leftarrow NIL$
$d[s] \leftarrow \infty$
for $i \leftarrow 1$ **to** $|V[G]| - 1$ **do**
    for each edge $(u, v) \in E[G]$ and $u \neq s$ **do**
        if $d[v] < min(d[u], p[u])$ **then**
            $d[v] \leftarrow min(d[u], p[u])$
            $\pi[v] \leftarrow u$
return $\pi[t]$

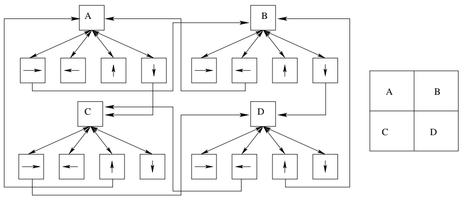

Figure 8: Four zones are in a square network field. The power of a zone is evaluated in four directions, left, right, up, and down. A zone is represented as a zone vertex with four direction vertices. The power labels are omitted from this figure.

5.3 Local Path Selection

Given a global route across zones, our goal is to find actual routes for messages within a zone. The *max-min* zP_{min} algorithm is used directly to route a message within a zone.

If there are multiple entry points into the zone, and multiple exit points to the next zone, it is possible that two paths through adjacent zones do not share any nodes. These paths have to be connected.

The following algorithm is used to ensure that the paths between adjacent zones are connected (see Figure 7 (right)). For each node in the overlap region, we compute how many paths can be routed locally through that node when zone power is evaluated. In order to optimize the message flow between zones, we find paths that go through the nodes that can sustain the maximal number of messages. Thus, to route a message through zone B in the direction from A to C we select the node with maximum message weight in the overlap between A and B, then we select the node with maximum message weight in the overlap between B and C, and compute the *max-min* zP_{min} paths between these two nodes.

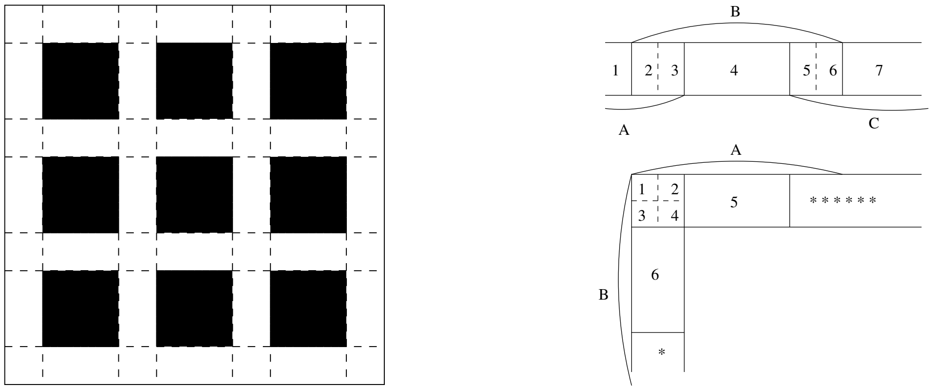

Figure 9: The scenario used for the zone-based experiment. The network space is a $10 * 10$ square with nine buildings blocking the network. Each building is of size $2 * 2$, and regularly placed at distance 1 from the others. The sensors are distributed randomly in the space nearby the buildings. Each sensor has an initial power of 4000. The power consumption formula is $e_{ij} = 10 * d_{ij}^3$. We partition the network space into 24 zones, each of which is of size $1 * 4$ or $4 * 1$, depending on its layout. For each zone, there is another corresponding zone with the same nodes but with opposite direction. For example, in the upper-right figure, areas $2, 3, 4, 5, 6$ constitute a zone, with 2 and 6 its source and sink areas; and $6, 5, 4, 3, 2$ constitute another zone with 6 and 2 its source and sink areas. We have a total of 48 zones. The right figures show the layout of the neighboring zones. In the upper figure, 3 is the sink area of the zone A, and 5 is the source area of zone C. The border area of A and B is $2, 3$; and the border area of B and C is $5, 6$. The lower figure shows two perpendicular zones. The source area of B is $1, 2$. The border area of A and B is $1, 2, 3, 4$.

5.4 Performance Evaluation for Zone-based Routing

The zone-based routing algorithm does not require as much information as would be required by *max-min* zP_{min} algorithm over the entire network. By giving up this information, we can expect the zone-based algorithm to perform worse than the *max-min* zP_{min} algorithm. We designed large experiments to measure how the zone-based algorithm does relative to the *max-min* zP_{min} algorithm. (In the following experiments, we only consider the power consumption used for the application messages instead of the control messages. Thus we can compare how much the performance of our zone-based algorithm is close to that of the *max-min* zP_{min} algorithm without the influence of the control messages.)

We disperse $1,000$ nodes randomly in a regular network space (see Figure 9). The zone partition is described in the figure. Each zone has averagely 40 nodes. Each node sends one message to a gateway node in each round (A round is the time for all the nodes to finish sending messages to the gateway). The zone power evaluation protocol is executed after each round. By running the *max-min* zP_{min} algorithm, we ran the algorithm for about 41000 messages before one of the hosts got saturated. By running the zone-based routing algorithm, we got about 39000 messages before the first message cannot be sent through. The performance ratio between the two algorithms in terms of the lifetime of the network is 94.5%. Without the zone structure, the number of control messages on the power of each node in every information update is 1000, and they need to be broadcast to 1000 nodes. In zone-based algorithm, the number of control messages is just the number of the zones, 48 here, and they are broadcast to 24 zones after the zone power evaluation. And the zone-based routing dramatically reduces the running time to find a route in our simulation. In another experiment, we disperse 1240 sensors to a square field with size $6.2 * 6.2$. The sensors are distributed randomly in the field. Each sensor has an initial power of 400. The power consumption formula is $e_{ij} = 10 * d_{ij}^3$. The network field is divided by 5*5 squares each of which corresponds to four zones in four directions (left, right, up and down). The zone-based algorithm achieved 96% of the lifetime of the *max-min* zP_{min} algorithm.

6 Distributed Power-aware Routing with *max-min* zP_{min}

The algorithms discussed in the previous sections do not work for applications where it is impossible to control in a centralized way the message flow in the ad-hoc network. Applications in which nodes move frequently and unpredictably fall in this category. In this section we investigate a class of routing algorithms for which computation is distributed and information localized.

We introduce three new algorithms: a distributed minimal power algorithm, a distributed max-min power algorithm, and the distributed *max-min* zP_{min} power-aware algorithm. The first two algorithms are used to define the third, although they are very interesting and useful in their own right for applications where the optimization criterion is the minimum power, respectively the maximum residual power.

6.1 A Distributed Minimal Power Algorithm

We can develop a distributed version of Dijkstra's algorithm that is guaranteed to be a minimal-power path, by giving messages variable propagation delays. The idea is to have messages traveling along short paths move faster than messages traveling along long paths. Thus, messages traveling along longer paths will arrive faster than messages traveling along longer paths—that is, the algorithm will select the shorter paths. In this case, the Dijkstra distance corresponds to power-consumption.

We can implement this idea by augmenting each message with a record of how far it traveled from the base to the current node. This information is represented by a variable attached to the message for the cost (distance representing power consumption). Algorithm 5 is the resulting minimal power path algorithm.

We continue this section by arguing that Algorithm 5 produces the minimal power-consumption path for each node. Furthermore, the running time of the algorithm is proportional to the longest shortest distance from the base node to any node.

We first examine a special case—when messages are time-sorted in the following sense. Suppose two messages carrying "distance" values v_1 and v_2 arrive at the same node at time t_1 and t_2. If for any two messages with $v_1 < v_2$, we have $t_1 < t_2$, the messages are *time-sorted.* Let n be the number of nodes in the network. In order to keep our proof simple, we assume that message transmission is instantaneous—this restriction can be relaxed.

Theorem 4 *If the messages are time-sorted, then Algorithm 5 requires $O(n)$ broadcasting messages ($O(1)$ for each node).*

Algorithm 5 *Minimal Power Path.* The input consists of a network system in which each node can determine its location and its power level. The output is the minimal-power routing table at each node (with respect to communicating to the base.) The algorithm uses the following parameters: η is the unit power for transforming the power level into waiting time; P_A is the total power consumption of the optimal path found so far from A to the base node; $e(A, B)$, the power consumption of sending one message from A to B directly; and t_B, the earliest time for B to broadcast the routing message.

1: **Initialization**;
2: Handshaking among neighbors; each node broadcasts its id, its position, and its current power level
3: $P_B = \infty$, $t_B = \infty$
4: **if** I am base station **then**
5:     initiate the message broadcasting
6: **else if** I am not base and my id is B **then**
7:     Receive message (A, P_A); get the sender id A and P_A from the message
8:     Compute $P_B = min(P_A + e(A, B), P_B)$ and $t_B = min(t_B, \eta P_B)$
9:     Wait till t_B, broadcast the message (B, P_B) to its neighbors, and stop

Proof: Let the message value of a message be the distance from the base station to the current node. Since the messages are time-sorted, the earliest message must carry the shortest distance from the base station to the current node. By line 9 of the algorithm, this message will be broadcast only once after the t_B waiting period has been completed. $\square$

In Algorithm 5 the messages are not time-sorted. However, the messages become time-sorted if we consider the broadcast time of a node as the message arrival time (because of the delays enforced by the algorithm) and by Theorem 4, Algorithm 5 gives the shortest path within $O(n)$ broadcasts.

Note that the performance of our algorithm depends on the granularity at which we can measure power. Let the smallest measurement unit for power consumption be s. The parameter η, which can be chosen as the smallest time unit a node can distinguish, is the waiting time that corresponds to distance s. The running time of Algorithm 5 is proportional to $1/s$ and to the size of the largest minimal power path. A large value for s results in a fast running time, but at the expense of precision. Say two messages that travel along paths with power consumption of P and $P + s_1$ (where $s_1 < s$) arrive at the same node in an interval less than η. The node may not distinguish them because the time difference is too small. Therefore, the running time is dependent on the precision of the required power consumption measurement. A better running time can be obtained by allowing a low measurement precision, that is, a large unit power consumption η. We can use these ideas to improve performance as described in Algorithm 6.

Let P be the maximal minimal power consumption from the base station to any node. We divide $[0, P)$ into m slots, $[0, P/m)$, $[P/m, 2P/m)$, $\cdots$, $[iP/m, (i+1)P/m)$, $\cdots$, $[(m-1)P/m, P)$. When a node receives a message with value v, it first finds the i^{th} slot such that $iP/m \leq v < (i+1)P/m$, waits till time $i\delta$, and then broadcasts the message to its neighbors. The running time of the algorithm ($m\delta$) is proportional to m and the parameter δ, which is the time interval corresponding to P/m.

We can choose δ to be large enough so that any message traveling from the base station to

Algorithm 6 The second minimal power path algorithm. The input is a network in which each node can determine its location and its power level. The output is a routing table for each node. The parameters are P_A, the total power consumption of the optimal path found so far from A to the base node; $e(A, B)$, the power consumption of sending a message from A to B directly; and δ, the unit time corresponding to each power slot (P/m), used to transform the power level into waiting time.

Initialization;
Handshaking among neighbors: each node broadcasts its id, its position, and its current power level
The base initiates the message broadcasting
if I am not the base **then**
  Let my id be B
  $P_B = \infty$. Initial time is 0.
  Receive message (A, P_A); get the sender id A and the power P_A from the message
  Compute the new power $P_B = min(P_B, P_A + e(A, B))$, and find the proper slot $i = \lfloor m \cdot P_B/P \rfloor$
  Set waiting timer to $i\delta$ (i.e. the time point when a broadcast happens)
  if the current time is no less than the waiting time point **then**
    broadcast the message (B, P_B) to its neighbors, and clear the timer. ; We do that because there are may be several paths being broadcast to the node. But their time must be between $i\delta$ and $(i + 1)\delta$
  if the current time is $(i + 1)\delta$ **then**
    stop

any node in the network along a minimal power path with total message processing time $\epsilon < \delta$. (That is, the sum of the message processing time at each node on the minimal power path is less than δ).

Theorem 5 *For Algorithm 6, the number of messages broadcast by each node is no greater than the maximal number of paths from the base to a node with the power consumption in the same slot as that of the minimal power path (that is, $[iP/m, (i+1)P/m)$ in which the minimal power consumption lies).*

Proof: Consider a message arriving at node A and scheduled to be broadcast in the slot $[i\delta, (i + 1)\delta)$.

The message traveling along the minimal power path arrives at A at some time point before $i\delta + \epsilon$ since we assume the total message handling time (including message buffering, queuing, and propagation) is less than ϵ.

A message traveling along a path with power no less than $(i + 1) \cdot \frac{P}{m}$ will not be scheduled to be broadcast because the node stops broadcasting at time $(i + 1)\delta$.

There is no path with power consumption less than $i \cdot \frac{P}{m}$ to that node, so no message can be broadcast before $i\delta$ by that node.

Thus, only the messages traveling along the paths with power in the range of $[P_{min}, (i+1)\delta)$ can be scheduled to broadcast. $\square$

Theorem 6 *Algorithm 6 gives the minimal power consumption route for each node.*

Proof:

The message traveling along the minimal power path arrives at A at some time point before $i\delta + \epsilon < (i+1)\delta$ since we assume the total message handling time (including message buffering, queuing, and propagation) is less than ϵ. There is no path with power consumption less than $i \cdot \frac{P}{m}$ to that node, so no message can be broadcast before $i\delta$ by that node.

Thus, the message traveling along the minimal power path will be broadcast at each node. Then each node can look at the power consumption value carried by the message and set the node who broadcast the message as its route. $\square$

6.2 A Distributed Max-Min algorithm

The minimal power path algorithm does not consider the residual power of nodes when computing the route. Although a packet is routed along the minimal power path, some nodes on that path may be saturated very quickly. An alternative is to use the nodes with high power and avoid the nodes that are almost saturated, which leads to the max-min path for packet routing.

The max-min path is defined as the route from a node to the base on which the minimal residual power of the nodes is maximized among all the routes. The minimal residual power of a path $p(c, d)$ is $c = a_1, a_2, \cdots, a_k = d$, defined as $m_{p(c,d)} = min_{i=1}^{n-1} \frac{P_{a_i} - e(a_i, a_{i+1})}{P_{a_i}}$. The max-min value is $F_{(c,d)} = max_{all\ p(c,d)} m_{p(c,d)}$. For multiple routes with the same max-min residual power, we can resolve ties arbitrarily.

Max-min paths can be found by using a modified version of the distributed Bellman-Ford algorithm. Upon computing a new max-min value, each node broadcasts it. The neighbors compute their max-min value according to the new incoming value, and broadcast the result only if the value is changed. The number of message broadcasts may be $O(n^3)$ as in the case of the distributed Bellman-Ford algorithm.

To reduce the message broadcasts, we employ the same method as in Section 6.1 and add a variable waiting time on each node, which controls when the node broadcasts. Algorithm 7 summarizes the resulting protocol. We assume all the nodes are synchronized well, so that they can decide locally the global time. Thus, a global clock is not needed to make this protocol work.

The max-min approximation, Algorithm 7 considers the maximal residual power fraction of all nodes in the network F_{max} split into m slots $([0, F_{max}/m), [F_{max}/m, 2F_{max}/m), \cdots, [iF_{max}/m, (i+1)F_{max}/m), \cdots, [(m-1)F_{max}/m, F_{max}))$. The m slots are mapped to consecutive δ long time slots $(s_1, s_2, \cdots, s_m.)$ In s_i the algorithm will find all the nodes whose max-min values are in slot $[(i-1)F_{max}/m, iF_{max}/m]$. The nodes found in the earlier slots have higher max-min values than those found in later slots.

We assume that the base has the maximal max-min value in the beginning of the algorithm. Thus, the base initiates the algorithm in the first slot s_1. Upon receiving the max-min values from the neighbors, nodes update their max-min value. Nodes wait until the time slot corresponding to the current max-min value, and then broadcast the value to their neighbors. If a node receives a new incoming value in some slot, say s_i, and finds that its max-min value should also be broadcast in this time slot, the broadcast is immediate. Thus, the nodes with max-min values in $[(i-1)F_{max}/m, iF_{max}/m)$ will be found as the messages go around the whole network.

If all the nodes have synchronized clocks, this algorithm performs $O(1)$ message broadcasts for each node. Otherwise, the base must initiate a synchronized broadcast to all the nodes to

Algorithm 7 Distributed Max-min Approximation. The input is a network in which each node can determine its location and its power level. The output is a routing table at each node. The parameters are: P_A, the total power consumption of the optimal path found so far from A to the base node; $e(A, B)$, the power consumption of sending one message from A to B directly; and δ, the unit time corresponding to each power slot (P/m) used to transform the power level into waiting time.

1: **Initialization**;
2: Handshaking among neighbors: each node broadcasts its id, its position, and its current power level
3: Each node B does the following for $i = m-1, m-2, \cdots, 1, 0$. $F_B = 0$
4: The base node initiates the search and broadcasts the maximal max-min value
5: **if** Node B receive a message (A, P_A, F_A) from its neighbor A **then**
6:    According to the power level of A and the distance between A and B, compute $F_B = max(F_B, min(F_A, \frac{P_A - e(A,B)}{P_A}))$
7:    **if** $F_B == min(F_A, \frac{P_A - e(A,B)}{P_A})$ **then**
8:       $N_B = A$
9: **if** $(i+1)F_{max}/m > F_B \geq iF_{max}/m$ **then**
10:    the max-min value of B is found
11:    B broadcasts the message (B, P_B, F_B), the next node in the routing table is A, stop
12: After time δ, i=i-1; go to 5

start a new slot and the number of broadcasts per node becomes $O(m)$.

Since each node broadcasts at most m messages, the running time of the algorithm is $m\delta$ where δ is the time for each round, which is at most n times the per message handling time. Furthermore, we can prove the following result using induction.

Theorem 7 *For each node, the algorithm gives a route with the minimal residual power fraction F, such that F and F^m are in the same slot where F^m is the max-min power fraction of the route from the base to that node. Then we have $|F - F^m| \leq F_{max}/m$.*

Proof: We use induction. In the first round, the maximal max-min value is broadcast by the base node. Each node that has the max-min value in the slot will broadcast the message.

For any node B with max-min value F_B^m in slot i, it is impossible for B to broadcast its value in slots before i. That is, F_B must be no greater than F_B^m, the actual max-min value of node B. This can be derived by examining the computation of F_B.

Suppose each node who finishes broadcast has F and F^m in the same slot. For any node B whose max-min value is in slot i, let A be the upstream node on the max-min path from the base to B. If B broadcasts its max-min value before A, then B can determine A's slot. Otherwise, A must broadcast its max-min value before B and B will hear the max-min value of A. Thus, from the algorithm, we have (see Algorithm 7) $min(F_A^m, \frac{P_A - e(A,B)}{P_A}) = F_B^m \geq F_B \geq min(F_A, \frac{P_A - e(A,B)}{P_A})$. We know $min(F_A^m, \frac{P_A - e(A,B)}{P_A})$ and $min(F_A, \frac{P_A - e(A,B)}{P_A})$ are in the same slot, so F_B and F_B^m are in the same slot. □

We can improve Algorithm 7 using binary search. The running time can be reduced to $\delta \log m$, but the number of total messages sent is $n \log m$. The key idea is to split the range $[0, F_{max})$ in two, $[0, F_{max}/2)$ and $[F_{max}/2, F_{max})$. In the first epoch, the algorithm tries to find all the nodes within the highest half max-min values. In the second epoch, we split each range into two halves to get four ranges. The algorithm finds in parallel all the nodes with highest half max-min values for each range, etc.

6.3 Distributed *max-min* zP_{min}

We now derive the distributed version of the centralized online *max-min* zP_{min} algorithm. Like in the centralized case, our motivation is to define a routing algorithm that optimizes the overall lifetime of the network by avoiding nodes of low power, while not using too much total power. There is a trade-off between minimizing the total power consumption and maximizing the minimal residual power of the network. We propose to enhance a max-min path by limiting its total power consumption.

Recall that the network is described as a graph in which each vertex corresponds to a node in the network, and only two nodes within the transmission ranges of each other have an edge connecting them in the graph. The power level of a node a is denoted as $P(a)$, and the power consumption to send a message unit to one of its neighbors b is denoted as $e(a, b)$. Let $s(a)$ be the power consumption for sending a unit message from a to the base station along the least power consumption path. Let $r(a)$ be the minimum residual power fraction of the nodes on a's *mmz* path. Let $f(a)$ be the power consumption along the *mmz* path.

An *mmz* path has the following properties:

1. it consists of two parts: the edge connecting a to one of its neighbors and the *mmz* path of that neighbor;

2. its total power consumption is less then or equal to $z \cdot s(a)$; and

3. among all those paths defined by (1) and (2), the max-min value of the *mmz* path is maximized.

More precisely, $p(a)$ the *mmz* path of node a, is: (1) a simple path from a to the base station; (2) $f(a) < z \cdot s(a)$; and (3) $p(a) = (a, b) \cup p(b)$, where b is a's neighbor such that for any other neighbor c $r(a) = min(r(b), \frac{P(a)-e(a,b)}{P(a)}) \geq min(r(c), \frac{P(a)-e(a,c)}{P(a)})$.

Theorem 8 *There is one node b_j such as $e(a, b_j) + f(b_j) \leq z \cdot s(a)$.*

Proof: Use induction. The case for base is obvious. Let b_j be the node on the shortest path from a to the base. $f(b_j) \leq z \cdot s(b_j)$ and $e(a, b_j) + s(b_j) = s(a)$. So $e(a, b_j) + f(b_j) \leq e(a, b_j) + z \cdot s(b_j) \leq z \cdot (e(a, b_j) + s(b_j)) = z \cdot s(a)$ □

Note that $s(a)$ can be computed easily by using $s(a) = min\ \{s(b) + e(a, b)\}$ where b is a's neighbor.

The definition of the *mmz* path actually gives a constructive method for computing incrementally the *mmz* path by keeping track of $s(node), r(node), p(node)$ of each node because the computation only depends on these values at the node's neighbors. Let $n(node)$ be the next node on the path $p(node)$. The resulting algorithm is shown as Algorithm 8. In the algorithm, the base station initiates the route exploration by broadcasting its route information ($s(base), r(base)$, and $n(base)$ to its neighbors). When a node's route information changes, it broadcasts its updated information. This broadcast triggers its neighbor nodes to check if their

Algorithm 8 Distributed *max-min* zP_{min}. The parameters are P^B_{min}, the minimal power consumption for node B to send a message to the base; P^B, the power consumption of the path discovered so far from the node to the base; P^B, node B's current power level; F_B, the maximal min residual power level of the found route to base from node B; and N^B: the next node on B's found route.

1: Find the minimal power consumption path for each node
2: The base node 0 initiates the route discovery
3: $P^0 = 0; F_0 = \infty; N^0 = 0$
4: Node 0 sends route discovery request to its neighbors
5: Each node B receives message from its neighbors $A_1, A_2, \cdots, A_k$
6: It waits for time δ, then compute: $P^B = min(P^{A_1} + e(B, A_1), P^{A_2} + e(B, A_2), \cdots, P^{A_k} + e(B, A_k))$ Find all the neighboring nodes such that $P^{A_i} + e(B, A_i) <= zP^{A_i}_{min}$ Among all those found neighbors, find the node with maximal $min(F_{A_k}, (P_B - e(B, A_k))/P_B)$ Let the node be N^B and the min value be F_B
7: Broadcast the P^B and F_B to its neighbors
8: Go to 5 until the routing table gets to equilibrium

route information changes. Every time the route information of a node changes the information is broadcast until the system achieves equilibrium.

In our distributed version of the Max-min zP_{min} algorithm, we expect a total of $O(n^3)$ message broadcasts in the worst case.

It is possible to improve the number of message broadcasts by using timing variables to suppress some of the messages. We can also vary the timing granularity by dividing into slots. Two specific approaches are

- In the max-min part, let the message carry the total power consumption on the path, and use the power consumption to decide if the max-min value should be accepted.
- In the minimal power path part, incorporate the max-min value in the waiting time.

6.4 Experiments in simulation

We have implemented the distributed algorithms outlined in this section and studied the performance of the distributed *max-min* zP_{min} algorithm. Furthermore, we compared this algorithm against a Greedy-style distributed algorithm.

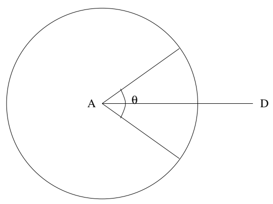

Figure 10: The greedy routing method sends messages the the nearest neighbor within transmission range, in a cone of directions captured by a parameter θ.

Figure 10 shows the concept behind our greedy routing implementation. Periodically, nodes exchange power information with their neighbors. When there is a message at A for destination D, A finds the node B with the highest power level in the its transmission range centered at A with angle θ, which is bisected by line AD, and sends the message to B.

Figure 11 shows the performance comparison of the distributed *max-min* zP_{min} algorithm and the distributed greedy algorithm. We conclude that *max-min* zP_{min} outperforms a simple greedy algorithm for all values of z, and for some values of z the distributed *max-min* zP_{min} doubles the performance. More specifically, peak of the *max-min* zP_{min} algorithm is obtained when z=1.2, and the number of messages sent is 29078. When z=2, the number message sent is the lowest at 18935. The distributed greedy algorithm sent 14278 messages in total. The performance improvement is 103% in the best case when z=1.2 and 32.61% in the worst case.

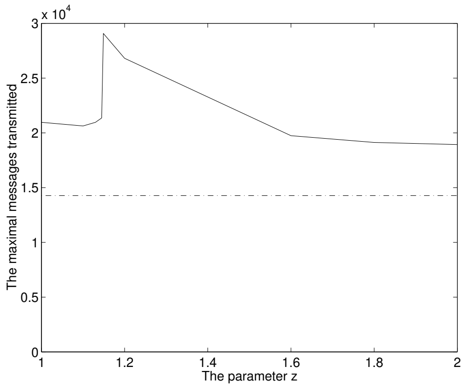

Figure 11: The performance comparison of distributed *max-min* zP_{min} algorithm and greedy algorithm. The dashed line shows the performance of the greedy algorithm and the solid line shows the performance of the *max-min* zP_{min} algorithm. The network includes 100 nodes. The network space is $100 * 100$, the transmission range is 20, the power consumption formula is $E = 2 * 10^{-6} * d^3$. The greedy algorithm uses a $\theta = \pi/3$. The routing protocol is run after every 100 messages. The neighbor information update in the greedy algorithm is updated every 100 messages.

We are currently collecting empirical data on the trade-offs between the various parameters we introduced to describe our algorithms.

7 Conclusion

We have described several online algorithms for power-aware routing of messages in large networks dispersed over large geographical areas. In most applications that involve ad-hoc networks made out of small hand-held computers, mobile computers, robots, or smart sensors, battery level is a real issue in the duration of the network. Power management can be done at two complementary levels (1) during communication and (2) during idle time. We believe that optimizing the performance of communication algorithms for power consumption and for the lifetime of the network is a very important problem.

It is hard to analyze the performance of online algorithms that do not rely on knowledge about the message arrival and distribution. This assumption is very important as in most real applications the message patterns are not known ahead of time. In this paper we have shown that it is impossible to design an on-line algorithm that has a constant competitive ratio to the optimal off-line algorithm, and we computed a bound on the lifetime of a network whose messages are routed according to this algorithm. These results are very encouraging.

We developed an online algorithm called the $max\text{-}min\ \ zP_{min}$ algorithm and showed that it had a good empirical competitive ratio to the optimal off-line algorithm that knows the message sequence. We also showed empirically that $max\text{-}min\ \ zP_{min}$ achieves over 80% of the optimal (where the optimal router knows all the messages ahead of time) for most instances and over 90% of the optimal for many problem instances. Since this algorithm requires accurate power values for all the nodes in the system at all times, we proposed a second algorithm which is hierarchical. Zone-based power-aware routing partitions the ad-hoc network into a small number of zones. Each zone can evaluate its power level with a fast protocol. These power estimates are then used as weights on the zones. A global path for each message is determined across zones. Within each zone, a local path for the message is computed so as to not decrease the power level of the zone too much. Finally, we have developed a distributed version of the $max\text{-}min\ \ zP_{min}$, in which all the decisions use local information only, and showed that this algorithm outperforms significantly a distributed greedy-style algorithm.

Acknowledgments. This work bas been supported in part by Department of Defense contract MURI F49620-97-1-0382 and DARPA contract F30602-98-2-0107, ONR grant N00014-01-1-0675, NSF CAREER award IRI-9624286, NSF award I1S-9912193, Honda corporation, and the Sloan foundation; we are grateful for this support. We thank Professor Ivan Stojmenovic for the suggestions on improving the paper.

References

[1] Adcon Telemetetry, http://www.adcon.com.

[2] Range LAN, http://www.proxim.com/products/rl2/7410.shtml.

[3] Jon Agre and Loren Clare. An integrated architeture for cooperative sensing networks. *Computer*, pages 106 – 108, May 2000.

[4] A.D. Amis, R. Prakash, T.H.P. Vuong, and D.T. Huynh. Max-min d-cluster formation in wireless ad hoc networks. In *Proceedings IEEE INFOCOM 2000. Conference on Computer Communications*, March 2000.

[5] Jae-Hwan Chang and Leandros Tassiulas. Energy conserving routing in wireless ad-hoc networks. In *Proc. IEEE INFOCOM*, Tel Aviv, Israel, Mar. 2000.

[6] Benjie Chen, Kyle Jamieson, Hari Balakrishnan, and Robert Morris. Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. In *7th Annual Int. Conf. Mobile Computing and Networking 2001*, Rome, Italy, July 2001.

[7] Yu Chen and Thomas C. Henderson. S-NETS: Smart sensor networks. In *Seventh International Symposium on Experiemental Robotics*, Hawaii, Dec. 2000.

[8] I. Chlamtac, C. Petrioli, and J. Redi. Energy-conserving access protocols for indetification networks. *IEEE/ACM Transactions on Networking*, 7(1):51–9, Feb. 1999.

[9] A. Chockalingam and M. Zorzi. Energy efficiency of media access protocols for mobile data networks. *IEEE Transactions on Communications*, 46(11):1418–21, Nov. 1998.

[10] B. Das, R. Sivakumar, and V. Bharghavan. Routing in ad hoc networks using a spine. In *Proceedings of Sixth International Conference on Computer Communications and Networks*, Sept. 1997.

[11] Deborah Estrin, Ramesh Govindan, John Heidemann, and Satish Kumar. Next century challenges: Scalable coordination in sensor networks. In *ACM MobiCom 99*, Seattle, USA, August 1999.

[12] Laura Maria Feeney and Martin Nilsson. Investigating the energy consumption of a wireless network interface in an ad hoc networking environment. In *INFOCOM 2001*, April 2001.

[13] M. Gerla, X. Hong, and G. Pei. Landmark routing for large ad hoc wireless networks. In *Proceedings of IEEE GLOBECOM 2000*, San Francisco, CA, Nov. 2000.

[14] Piyush Gupta and P. R. Kumar. Critical power for asymptotic connectivity in wireless networks. *Stochastic Analysis, Control, Optimization and Applications: A Volume in Honor of W.H. Fleming*, pages 547–566, 1998.

[15] Z. J. Haas. A new routing protocol for the reconfigurable wireless network. In *Proceedings of the 1997 IEEE 6th International Conference on Universal Personal Communications, ICUPC'97*, pages 562 –566, San Diego, CA, October 1997.

[16] W. Rabiner Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient routing protocols for wireless microsensor networks. In *Hawaii International Conference on System Sciences (HICSS '00)*, Jan. 2000.

[17] Chalermek Intanagonwiwat, Ramesh Govindan, and Deborah Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *Proc. of the Sixth Annual International Conference on Mobile Computing and Networks (MobiCOM 2000)*, Boston, Massachusetts, August 2000.

[18] Mario Joa-Ng and I-Tai Lu. A peer-to-peer zone-based two-level link state routing for mobile ad hoc networks. *IEEE Journal on Selected Areas in Communications*, 17, Aug. 1999.

[19] D. B. Johnson and D. A. Maltz. Dynamic source routing in ad-hoc wireless networks. In T. Imielinski and H. Korth, editors, *Mobile Computing*, pages 153 –181. Kluwer Academic Publishers, 1996.

[20] Y. B. Ko and N. H. Vaidya. Location-aided routing (LAR) in mobile ad hoc networks. In *Proceedings of ACM/IEEE MOBICOM'98*, pages 66 – 75, 1998.

[21] P. Krishna, N.H. Vaidya, M. Chatterjee, and D.K. Pradhan. A cluster-based approach for routing in dynamic networks. *Computer Communication Review*, 27, April 1997.

[22] Qun Li, Javed Aslam, and Daniela Rus. Online power-aware routing in wireless ad-hoc networks. In *MOBICOM*, pages 97–107, Rome, July 2001.

[23] A.B. McDonald and T.F. Znati. A mobility-based framework for adaptive clustering in wireless ad hoc networks. *IEEE Journal on Selected Areas in Communications*, 17, Aug. 1999.

[24] S. Murthy and J. J. Garcia-Luna-Aceves. An efficient routing protocol for wireless networks. *ACM/Baltzer Journal on Mobile Networks and Applications*, MANET(1,2):183 –197, October 1996.

[25] V. Park and M. S. Corson. A highly adaptive distributed algorithm for mobile wireless networks. In *Proceedings of INFOCOM'97*, Kobe, Japan, April 1997.

[26] M.R. Pearlman and Z.J. Haas. Determining the optimal configuration for the zone routing protocol. *IEEE Journal on Selected Areas in Communications*, 17, Aug. 1999.

[27] C. E. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. *Computer Communication review*, 24(4):234 –244, October 1994.

[28] G. J. Pottie and W. J. Kaiser. Wireless integrated newtork sensors. *Communications of the ACM*, 43(5):51–58, May 2000.

[29] S. Ramanathan and M. Steenstrup. Hierarchically-organized, multihop mobile networks for multimedia support. *ACM/Baltzer Mobile Networks and Applications*, 3(1):101–119, June 1998.

[30] Volkan Rodoplu and Teresa H. Meng. Minimum energy mobile wireless networks. In *Proc. of the 1998 IEEE International Conference on Communications, ICC'98*, volume 3, pages 1633–1639, Atlanda, GA, June 1998.

[31] Elizabeth Royer and C-K. Toh. A review of current routing protocols for ad hoc mobile wireless networks. In *IEEE Personal Communication*, volume 6, pages 46 – 55, April 1999.

[32] S. Singh, M. Woo, and C. S. Raghavendra. Power-aware routing in mobile ad-hoc networks. In *Proc. of Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking*, pages 181–190, Dallas, TX, Oct. 1998.

[33] Ivan Stojmenovic and Xu Lin. Power aware localized routing in wireless networks. *IEEE Transactions on Parallel and Distributed Systems*, 12(11):1122–1133, November 2001.

[34] Ivan Stojmenovic, Mahtab Seddigh, and Jovisa Zunic. Dominating sets and neighbor elimination-based broadcasting algorithms in wireless networks. *IEEE Transactions on Parallel and Distributed Systems*, 13(1):14–25, January 2002.

[35] J. Wu, F. Dai, M. Gao, and I. Stojmenovic. On calculating power-aware connected dominating set for efficient routing in ad hoc wireless networks. *IEEE/KICS Journal of Communications and Networks*, 4(1):59–70, March 2002.

[36] J. Wu and H. Li. A dominating-set-based routing scheme in ad hoc wireless networks. *Telecommunication Systems Journal*, 3:63–84, 2001.

[37] J. Wu, B. Wu, and I. Stojmenovic. Power-aware broadcasting and activity scheduling in ad hoc wireless networks using connected dominating sets. In *IASTED International Conference on Wireless and Optical Communication*, Banff, Canada, July 2002.

[38] Ya Xu, John Heidemann, and Deborah Estrin. Adaptive energy-conserving routing for multihop ad hoc networks. *Research Report 527 USC/Information Sciences Institute*, October 2000.

[39] Wei Ye, John Heidemann, and Deborah Estrin. An energy-efficient mac protocol for wireless sensor networks. In *INFOCOM*, New York, NY, June 2002.