Real-time Power-Aware Routing in Sensor Networks

†Octav Chipara, ‡ Zhimin He, †Guoliang Xing, ‡Qin Chen, †Xiaorui Wang,
†Chenyang Lu, ‡John Stankovic, *Tarek Abdelzaher
† Washington University in St. Louis {ochipara, xing, wang, lu}@cse.wustl.edu
‡ University of Virginia {zh5f, qc5y, stankovic}@cs.virginia.edu
*University of Illinois at Urbana-Champaign zaher@cs.uiuc.edu

***Abstract*— Many wireless sensor network applications must resolve the inherent conflict between energy efficient communication and the need to achieve desired quality of service such as end-to-end communication delay. To address this challenge, we propose the *Real-time Power-Aware Routing* (RPAR) protocol, which achieves application-specified communication delays at low energy cost by dynamically adapting transmission power and routing decisions. RPAR features a power-aware forwarding policy and an efficient neighborhood manager that are optimized for resource-constrained wireless sensors. Moreover, RPAR addresses important practical issues in wireless sensor networks, including lossy links, scalability, and severe memory and bandwidth constraints. Simulations based on a realistic radio model of MICA2 motes show that RPAR significantly reduces the number of deadlines missed and energy consumption compared to existing real-time and energy-efficient routing protocols.**

I. Introduction

Many wireless sensor network (WSN) applications require real-time communication. For example, a surveillance system needs to alert authorities of an intruder within a few seconds of detection [1]. Similarly, a fire-fighter may rely on timely temperature updates to remain aware of current fire conditions [2]. Supporting real-time communication in WSNs is very challenging. First, WSNs have lossy links that are greatly affected by environmental factors [3][4]. As a result, communication delays are highly unpredictable. Second, many WSN applications (e.g., border surveillance) must operate for months without wired power supplies. Therefore, WSNs must meet the delay requirements at minimum energy cost. Third, different packets may have different delay requirements. For instance, authorities need to be notified sooner about high-speed motor vehicles than slow-moving pedestrians. To support such applications, a real-time communication protocol must adapt its behavior based on packet deadlines. Finally, due to the resource constraints of WSN platforms, a WSN protocol should introduce minimal overhead in terms of communication and energy consumption and use only a fraction of the available memory for its state.

To address these challenges, we propose the *Real-time Power-Aware Routing (RPAR)* protocol, which supports energy-efficient real-time communication in WSNs. RPAR achieves this by dynamically adapting transmission power and routing decisions based on packet deadlines. RPAR has several salient features. First, it improves the number of packets meeting their deadlines at low energy cost. Second, it has an efficient neighborhood manager that quickly discovers forwarding choices (pairs of a neighbor and a transmission power) that meet packet deadlines while introducing low communication and energy overhead. Moreover, RPAR addresses important practical issues in WSNs, including lossy links, scalability, and severe memory and bandwidth constraints.

In the rest of the paper, we first analyze the impact of transmission power on communication delay via an empirical study (Section II) and identify the design goals for real-time power-aware routing (Section III). Next, we present the design of RPAR (Section IV). We evaluate the performance of RPAR through simulations based on a realistic radio model (Section V). We conclude the paper with discussions on open issues (Section VI) and related work (Section VII).

II. Impact of Transmission Power on Delay

RPAR is based on the hypothesis that there is an inherent tradeoff between transmission power and communication delay. In this section, we study the impact of transmission power on communication delay in WSNs. We first quantify their relationship through experiments on XSM2 motes. We then discuss the tradeoff between communication delay and network capacity.

A. Empirical Study on XSM2 Motes

To understand the impact of transmission power on end-to-end communication delay, we perform a set of experiments in an office environment using XSM2 motes. Each XSM2 mote is equipped with a Chipcon CC1000 radio. The bandwidth of the radio is 38.4 Kbps, but the effective bandwidth is lower due to packet loss. Five XSM2 motes are placed in a line. The first mote injects packets into the network at a rate of 4 packets per second. Each mote forwards a packet to its next neighbor. When a packet reaches the end of the line, the last mote changes the packet's direction and sends it back to the source. Each mote runs TinyOS with B-MAC [5] as the MAC protocol. We implemented the Automatic Repeat Request (ARQ) mechanism to improve reliability. Each packet is transmitted at most 5 times. The data and acknowledgement packets are transmitted at the same transmission power. The transmission power is varied from -18 dbm to 2 dbm in increments of 1 dbm. The one-hop distance is varied from 5 feet to 40 feet, in increments of 5 feet. 100 packets are sent at each power level.

To evaluate the impact of transmission power on end-to-end delay, we measure the *delivery velocity* of each packet. The delivery velocity is defined as the distance a packet travels divided by its end-to-end delay. As shown in Figure 1, transmission power has a significant impact on delivery velocity. For example, when the one-hop distance is 20 feet,

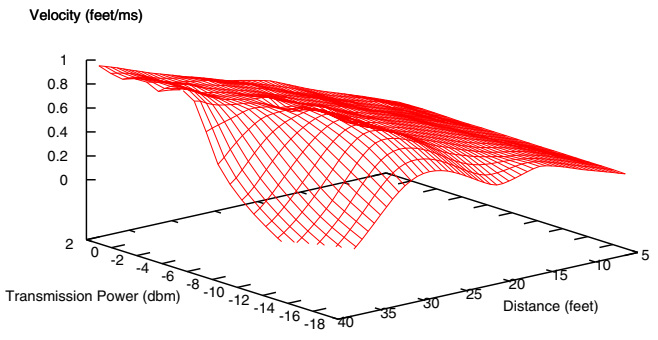


Fig. 1. Impact of transmission power and one-hop distance on delivery velocity.

increasing the transmission power results in more than a two-fold improvement in delivery velocity, from 0.25 feet/ms at -18 dbm to 0.54 feet/ms at 2 dbm. This is because increasing transmission power effectively improves link quality [6] and, therefore, reduces the number of transmissions needed to deliver a packet. This shows that under light workloads, poor link quality is the root cause of long delays. At each power level, the delivery velocity increases as the one-hop distance increases within a range but drops sharply when the one-hop distance exceeds the range due to degrading link quality. The initial improvement in the delivery velocity is due to the packet traveling longer distances at each hop. The drop-off range of delivery velocity corresponds to the boundary of the gray area in packet reception ratio reported in [3][7]. A higher transmission power results in a longer drop-off range, e.g., the neighbor located at 40 feet is not in the communication range when the transmission power is -18 dbm but it has good link quality and high delivery velocity at 2 dbm. Therefore, distant nodes with poor quality links at low power are transformed into reliable communication neighbors when the transmission power is increased, achieving high delivery velocities.

Our experiments demonstrate that transmission power control may be an effective mechanism for controlling communication delays under light workloads. In the following subsection, we discuss the tradeoff between communication delays and network capacity under heavier workloads.

B. Tradeoff between Delay and Capacity

Increasing transmission power has the side effect of reducing the maximum achievable throughput in a WSN due to increased channel contention and interference [8]. Our focus is on real-time applications in which meeting the deadlines of critical data is more important than the total throughput. For example, in a surveillance application, timely delivery of the location of an intruder is more important to the user than delivering a large amount of non-critical data.

It is also important to note that the reduced capacity is a problem only when the workload approaches the network capacity. Recent advances in real-time capacity theory show that the performance degradation may be avoided as long as the amount of high-priority data transmitted in the network is small enough not to trigger capacity bottlenecks. In [9], the authors derive a lower bound on the maximum amount of real-time traffic that may be transmitted without triggering capacity bottlenecks. This bound may be used to perform off-line analysis of capacity requirements or, on-line, for admission control or congestion control. We discuss how to integrate RPAR with such techniques in Section VI-A.

RPAR achieves the desired tradeoff among communication delay, energy consumption, and network capacity by adapting the transmission power based on required communication delays. When deadlines are tight, RPAR trades capacity and energy for shorter communication delay by increasing the transmission power. Conversely, when the deadlines are loose, RPAR lowers the transmission power to increase throughput and reduce energy consumption. This adaptive approach is a key feature of RPAR.

III. Problem Formulation

Due to the unreliable and dynamic nature of WSNs, it is unrealistic to provide hard delay guarantees. RPAR assumes that each packet is assigned a soft deadline by the application, which specifies the desired bound on the end-to-end delay of a packet. The primary goal of RPAR is to increase the number of packets that meet their deadlines while minimizing the energy consumed for transmitting packets under their deadline constraints. RPAR focuses on minimizing the energy consumed in packet transmissions. In addition, RPAR is designed based on the following principles:

- WSN applications have varied communication requirements resulting in workloads with diverse deadlines. A real-time power-aware routing protocol should dynamically adapt its transmission power and routing decisions based on workload and packet deadlines.
- The design of RPAR should account for the realistic characteristics of WSNs including loss links [3] and extreme resource constraints in terms of memory, bandwidth and energy.
- RPAR should be localized protocol that makes decisions based solely on one-hop neighborhood information. This property enables RPAR to scale effectively to large WSNs.

In this paper, we assume that each node is stationary and knows its location via GPS or other localization services [10]. Localization is a basic service essential to many applications that need to know the physical location of sensor readings. We also assume that radios can adjust their transmission power. For example, the Chipcon CC1000 radio can vary its transmission power between -20 dbm and 10 dbm. RPAR is designed to work with existing simple CSMA/CA protocols such as the B-MAC protocol [5] in TinyOS. To be consistent with the default configuration of B-MAC, RPAR assumes that the MAC protocol does not use RTS/CTS. RPAR may be easily extended to work with MACs that use RTS/CTS.

IV. Design of RPAR

RPAR has four components: a dynamic velocity assignment policy, a delay estimator, a forwarding policy, and a neighborhood manager. RPAR uses the velocity assignment policy to map a packet's deadline to a required velocity. The delay estimator evaluates the one-hop delay of each *forwarding choice* (N, p) in the neighbor table, i.e. the time it takes a node to deliver a packet to neighbor N at power level p. Based on the velocity requirement and the information provided by the delay estimator, RPAR forwards the packet using the most energy-efficient forwarding choice in its neighborhood table that meets the required velocity. When the forwarding policy cannot find a forwarding choice that meets the required velocity in the neighbor table, the neighborhood manager attempts to find a new forwarding choice that meets the required velocity through power adaptation and neighbor discovery.

A. Dynamic Velocity Assignment Policy

Before a node S forwards a packet, it uses the velocity assignment policy to compute the *required velocity* based on the progress made toward the destination and the packet's *slack*. The slack is the time remaining until the packet deadline expires and is part of the packet header. The application running on the source node initializes the slack with the (relative) deadline. The slack is updated at each hop to account for the queueing, contention, and transmission delays. To measure the queuing delay node S time-stamps the packet when it is received ($t_{rec}(S)$) and when it becomes the head of the transmission queue ($t_{head}(S)$). Let $slack_{rec}(S)$ be the slack of the packet when S receives it. We account for the queueing delay by subtracting it from the slack, i.e., $slack(S) = slack_{rec}(S) - (t_{head}(S) - t_{rec}(S))$. The required velocity of a packet when it becomes the head of the transmission queue is:

$$v_{req}(S, D) = \frac{d(S, D)}{slack_{rec}(S) - (t_{head}(S) - t_{rec}(S))} \quad (1)$$

where, D is the destination of the packet and $d(S, D)$ is the Euclidean distance between S and D. It is important to note that the deadline is met if the required velocity is met at each hop. Hence, RPAR maps the problem of meeting end-to-end deadlines to the local problem of meeting the required velocity at each hop.

When a node acquires the channel and is about to transmit a packet, it updates the slack in the packet header to account for the contention and transmission delays before transmitting it. The slack are also updated to account for the additional delays before each retransmission caused by packet loss. Note that updating the slack does not require clock synchronization.

This dynamic velocity assignment policy adapts the packet's required velocity based on current network conditions. If a packet is late, then its required velocity is increased so that it may catch up. Conversely, if the packet is early, its required velocity is decreased. In addition, the velocity assignment policy is used to prioritize the packets in the transmission queue according to their required velocity, with packets having a higher required velocity being transmitted first.

B. Forwarding Policy

RPAR makes forwarding decisions on packet-by-packet basis. In the following discussion we assume that RPAR forwards the packet that is the head of the transmission queue on node S and is destined for node D. RPAR forwards the packet to the most energy-efficient forwarding choice that meets the packet's required velocity. The velocity provided by (N, p) is:

$$v_{prov}(S, D, (N, p)) = \frac{d(S, D) - d(N, D)}{delay(S, (N, p))} \quad (2)$$

The progress made toward the destination by forwarding the packet to N is $d(S, D) - d(N, D)$. The estimated delay of forwarding choice (N, p), $delay(S, (N, p))$, approximates the time interval from when a packet becomes the head of the transmission queue until it is received at the next hop. The estimate is computed using the delay estimator (described in Section IV-C). Note that the delay estimator does not consider the queuing delay as it is already accounted for in the dynamic velocity assignment policy (as discussed in Section IV-A). The estimated one-hop delay is used to determine if a forwarding choice is eligible. A forwarding choice (N, p) is an *eligible forwarding choice* if the velocity it provides $v_{prov}(S, D, (N, p))$ is higher than the packet's required velocity $v_{req}(S, D)$.

RPAR then estimates the energy cost of all eligible forwarding choices. It uses the following formula to approximate the energy consumption of routing a packet from the current node S to its destination D through forwarding choice (N, p):[1]

$$E(S, D, (N, p)) = E(p) \cdot R(S, (N, p)) \cdot \frac{d(S, D)}{d(S, D) - d(N, D)} \quad (3)$$

where $E(p)$ is the energy consumed for transmitting a packet at power level p. $R(S, (N, p))$ is the expected number of transmissions before S successfully delivers a packet to N when transmitting at power level p. R is computed by the delay estimator. $d(S, D) - d(N, D)$ represents the progress towards D when N is selected as next-hop.

C. Delay Estimator

The delay estimator is responsible for estimating the delay of different forwarding choices. The delay of a packet sent by S to neighbor N using transmission power p depends on the contention delay $delay_{cont}(S)$ (i.e., the time to acquire the channel), the total transmission time of the packet and its acknowledgement $delay_{tran}$, and the transmission count $R(S, (N, p))$[2]:

[1] Equation (3) resembles the routing metric proposed in [11], which outperformed greedy geographic routing when a *fixed* transmission power is used. Our forwarding policy extends this metric to estimate the energy cost of forwarding choices with *different* power levels.

[2] In case of a failed transmission, the sender waits for the transmission time of acknowledgement before retransmitting the packet. The data and acknowledgment packets are sent at the same power level. The propagation delay is ignored, as WSN usually use short-range radios.

$$delay(S,(N,p)) = (delay_{cont}(S) + delay_{tran}) \cdot R(S,(N,p)) \quad (4)$$

Since the total transmission time of a packet and its acknowledgement is a constant determined by packet size and network bandwidth, the main function of the delay estimator is to predict the contention delay and the number of transmissions required to successfully forward a packet to a neighbor. Since the contention delay is independent of the forwarding choice when RTS/CTS is not used, our delay estimator consists of a single contention estimator *per node* and a transmission count estimate *per forwarding choice*. This reduces the storage cost of the delay estimator.

Our delay estimator is designed to support real-time communication in dynamic environments. Existing link estimators are designed to estimate the *average* link quality [7][12]. These approaches are ill-suited for real-time communication since routing decisions based on average delays may result in a large number of deadline misses due to high variability in communication delays. In contrast, our delay estimator adapts Jacobson's algorithm [13] to calculate conservative estimations of contention delays and transmission counts. Jacobson's algorithm was originally used in TCP to compute a retransmission timeout based on round-trip times between the source and the destination of a TCP flow. The retransmission timeout is computed by adding to the average round trip time its variation multiplied by a scaling term. We adopt Jacobson's algorithm because it considers both the average and variation of the estimated variable and, as a result, provides a better estimate of its worst-case value. This enables us to reduce the number of deadlines misses. Similarly, we compute a conservative estimate of the transmission count for each forwarding choice by considering the average and variation in the observed transmission count. However, if a packet is dropped after exceeding the allowed number of transmissions (according to ARQ), the transmission count estimate is set to infinity. A conservative estimate of the contention delay is also computed based on the average and variation of the observed contention delays. Equation (4) is used to estimate the delay of a packet by combining the estimated transmission count and estimated contention delay.

D. Neighborhood Manager

A key challenge for RPAR is to quickly discover eligible forwarding choices that are energy-efficient. This is particularly challenging because a node needs to select among a large number of forwarding candidates of which only a few may meet the velocity requirements. In WSN, due to the probabilistic nature of wireless links [7], a node hears transmissions from a potentially large number of neighbors including those that have poor link quality and long delays. In addition, wireless interfaces typically have a large number of power levels to support fine-grained power control. For example, XSM2 motes support 31 power levels. Unfortunately, typical WSN platforms have limited memory allowing us to store only a few forwarding choices. For example, MICA2 motes have only 4KBs RAM, of which a small fraction may be dedicated to neighborhood management. More importantly, empirical studies show that link quality is highly variable over time [3][4][7]. As a result, the estimated delay and energy consumption of the forwarding choices that are used infrequently become outdated and inaccurate. Using this information may result in increased energy consumption or, even worse, in deadline misses. Therefore information about such forwarding choices must be refreshed when needed. This makes efficient discovery of eligible forwarding choices an important problem even on platforms without severe memory constraints.

RPAR features a novel neighborhood manager that dynamically discovers eligible forwarding choices and manages the neighborhood table. The neighborhood manager is invoked whenever there are no eligible forwarding choices in the neighbor table for forwarding a packet. It supports two mechanisms for discovering new forwarding choices: adapting the transmission power to a neighbor already present in the neighbor table (*power adaptation*) or discovering new neighbors (*neighbor discovery*).

1) Power Adaptation : When the required velocity of a packet cannot be satisfied, the power adaptation scheme increases the transmission power to improve the velocity provided by neighbors already in the neighbor table. Conversely, when velocity requirements are satisfied by known forwarding choices, it attempts to improve energy efficiency by decreasing the transmission power. RPAR adjusts the transmission power to a neighbor using a multiplicative increase and linear decrease scheme as discussed below.

When the forwarding policy cannot find an eligible forwarding choice in the neighbor table, RPAR determines a neighbor whose power should be increased to achieve higher delivery velocity. A node is *eligible for power increase* if its transmission count may be reduced by increasing the transmission power. A node becomes ineligible for power increase if (1) the maximum transmission power is reached or (2) the estimated transmission count is one (i.e., the link quality is perfect). RPAR chooses the neighbor with the maximum velocity among all neighbors eligible for power increase, and multiplies the transmission power to that neighbor by a factor α ($\alpha > 1$). If RPAR cannot find a neighbor eligible for power increase, it invokes neighbor discovery to find new neighbors (see Section IV-D.2).

The power adaptation scheme may also decrease the transmission power to improve energy efficiency and network capacity. When the neighbor table contains forwarding choices that meet the velocity requirement of incoming packets, RPAR decreases the power of the most energy-efficient forwarding choice by β ($\beta > 0$) until one of the following conditions is satisfied: (1) the minimum power has been reached, (2) the estimated transmission count exceeds the number of allowed retransmission, or (3) there are two consecutive power levels such that at the lower level the required velocity is not met but at the higher power level the required velocity is met.

A large α reduces the time it takes to reach sufficient

power to find an eligible forwarding choice. However, it may waste energy when a lower transmission power may suffice for meeting the required velocity. A large β allows RPAR to quickly reduce the power but it may also result in deadline misses when the power is reduced too aggressively.

The power adaptation scheme provides a responsive mechanism for adapting to variations in link quality. A key benefit of this scheme is that it does not introduce any overhead packets.

2) Neighbor Discovery : When RPAR cannot find an eligible forwarding choice through power adaptation, it uses the neighbor discovery component to find new neighbors that can meet the required velocity.

In the following discussion, we assume that S routes a packet to D and no eligible forwarding choice that meets the required velocity v_{req} exists in S's neighbor table. S starts neighbor discovery by broadcasting a Request To Route (RTR) packet at some power p. Some node N hears the RTR and replies. Upon receiving the reply, node S inserts in its neighbor table the new forwarding choice (N, p). We need to address three issues: (1) What is the transmission power p at which an RTR is transmitted? (2) How can we minimize the communication overhead for neighborhood discovery? (3) How can we reduce the time it takes to find a neighbor that meets the required velocity?

When neighbor discovery is triggered because there is no neighbor closer to the destination in the neighbor table, S broadcasts an RTR at the medium power level. This usually occurs when a node routes a packet to a new destination. We chose to transmit the RTR at the medium power level to reduce the impact of neighbor discovery on network capacity and energy consumption. In contrast, if a neighbor closer to the destination is in the neighbor table, the RTR is broadcast at the maximum power. This ensures that far away nodes that may provide high delivery velocities receive the RTR.

Since the RTR is broadcast, a large number of nodes may reply and cause severe network contention. This problem can be mitigated by requiring each node to wait for a random interval before it is allowed to reply. A node does not reply if it hears replies from other nodes. However, This simple scheme has a drawback: while a large time window reduces the chance of packet collisions, it prolongs the time needed to find an eligible neighbor. It is crucial to discover new forwarding choices quickly, since neighbor discovery time is part of the end-to-end delay and therefore may lead to deadline misses[3].

To find a new neighbor, our neighborhood manager restricts the set of replying nodes to those that can meet the required velocity. A node replies only if it satisfies the following conditions: (1) it makes progress toward the destination, (2) it is not already in S's neighbor table, and (3) the maximum velocity that may be achieved by selecting it as next hop is higher than the required velocity. To verify that a node makes progress to the destination, we include the sender and destination locations in the RTR. In addition, the RTR may piggyback a list of node IDs which are in the table and, hence, should not reply (within the limit of packet size). A neighbor N replies if the following inequality is satisfied:

$$v_{req}(S,D) \leq v_{max}(S,D,N) = \frac{d(S,D) - d(N,D)}{delay_{cont}(S) + delay_{tran}} \quad (5)$$

where $v_{max}(S,D,N)$ is the maximum velocity that N can provide to a packet being routed from S to D. v_{max} is computed based on the minimum possible delay which occurs when the transmission count between S and N is one ($R(S,(N,p)) = 1$). From (5), the maximum distance between any eligible neighbor N and destination D can be derived as follows:

$$d_{max}(N,D) = d(S,D) - v_{req}(S,D) \cdot (delay_{cont}(S) + delay_{tran}) \quad (6)$$

S piggybacks $d_{max}(N,D)$ in the RTR, and a neighbor N that hears the RTR replies only if $d(N,D) \leq d_{max}(N,D)$.

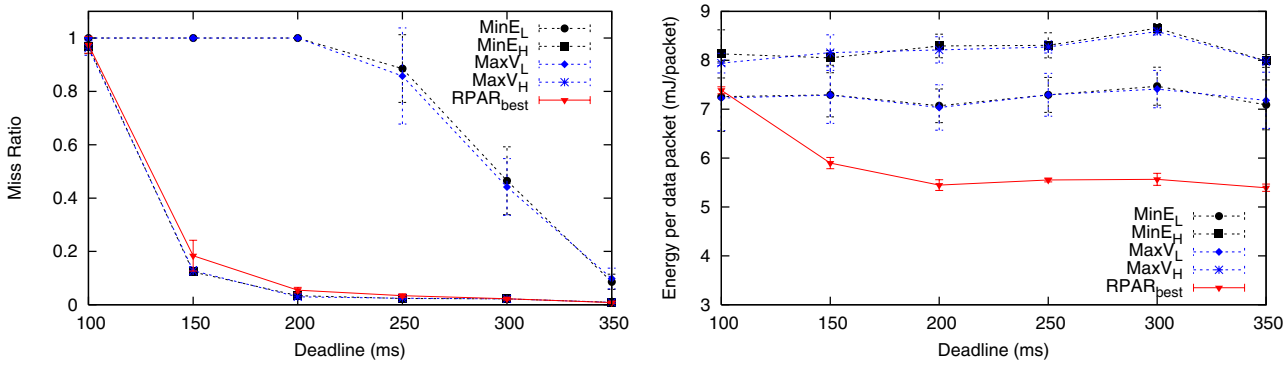
3) Neighborhood Table Management: In contrast to earlier neighborhood management techniques that rely on periodic beacons [7], our power adaptation and neighbor discovery schemes are triggered *on-demand*, when no eligible forwarding choices exist in the neighbor table. The reactive approach enables our neighborhood manager to respond quickly to changes in the network conditions and packet deadlines while introducing low overhead when network and workload remain unchanged.

Similar to MT [7], we use the FREQUENCY algorithm [14] to manage the neighbor table. The FREQUENCY algorithm associates a frequency counter with each forwarding choice. When a forwarding choice is used for routing, its frequency counter is incremented while the frequency counters of all other forwarding choices are decremented. When the neighbor manager inserts a new forwarding choice and the table is full, the forwarding choice with the smallest frequency count is evicted. Since only the frequently used forwarding choices remain in the table, RPAR adapts the set of known forwarding choices to the velocity requirements of the incoming packets. To avoid using stale data in larger neighbor tables, we add a timeout value to each forwarding choice which is reset upon using the forwarding choice. Forwarding choices that exceed the timeout value are considered stale and are evicted.

V. Experimental Evaluation

We implement RPAR in a Matlab-based network simulator called Prowler [15]. To create a realistic simulation environment, we configure Prowler based on the characteristics of MICA2 motes [16], which share the same hardware configurations as the XSM2 motes but with different packaging. A node can transmit packets at 31 power levels, ranging from -20 dBm to 10 dBm, with current consumption from 3.7 mA to 21.5 mA. The bandwidth is 40 Kbps. Prowler uses a log-normal shadowing path-loss propagation model at the physical layer. A collision occurs if a node receives two overlapping packets with signal strengths over the receiver's sensitivity. We implement the probabilistic link model from USC [17]

[3]RPAR accounts for the delay caused by power adaptation and neighbor discovery by subtracting it from the slack.



(a) Miss ratio when the deadline is varied

(b) Energy per data packet when the deadline is varied

Fig. 2. Performance of considered protocols when deadline is varied. The neighborhood table is prefilled.

in Prowler. Experimental data shows that the USC model produces lossy and asymmetric links similar to MICA2 motes [6]. The MAC protocol in Prowler employs a simple CSMA scheme similar to B-MAC, TinyOS's MAC protocol [5]. To improve the reliability, we use ARQ with a maximum number of five transmissions per packet. The size of the data and acknowledgment packets are 760 and 200 bits, respectively.

We evaluate RPAR's real-time performance and energy efficiency using the following performance metrics: *miss ratio*, defined as the fraction of packets that missed their deadlines, and the *energy per data packet*. The energy per packet is the total transmission energy consumed in a run divided by the number of data packets successfully delivered to their destinations. We compare RPAR with two protocols that consider real-time or energy-efficient communication. The first baseline protocol, MaxV, is inspired by SPEED [12], which supports soft real-time communication by enforcing a uniform delivery velocity across the network. However, to reduce the delay MaxV, always chooses the neighbor with the maximum velocity. The second baseline, MinE, is an energy-efficient geographic routing protocol that selects as next hop the most energy efficient forwarding choice according to Equation (3). This routing scheme significantly outperforms greedy geographic routing in terms of energy efficiency in lossy wireless networks [11]. Unlike RPAR, these baseline protocols operate at a fixed transmission power level. We use $protocol_L$ and $protocol_H$ to denote the protocol (MaxV or MinE) that operates at the default power (0 dBm) and the maximum power (10 dBm), respectively.

In simulations, we focus on the "many-to-one" traffic pattern that is common in WSN applications. In each simulation, 130 nodes are deployed in a 150 m $\times$ 150 m region divided into 11.5 m $\times$ 15 m grids. A node is randomly positioned in each grid. To increase the hop count between sources and the sink, we choose sources from the left-most grids of the topology. The sink is located in the middle of the right-most grids. A source sets the interval between sending two consecutive packets to be the sum of a constant (300ms) and a random value that follows an exponential distribution. We vary the mean of the exponential distribution to create different workloads. Each result is the average of five runs. The 90% confidence interval of each data point is also presented.

We start by evaluating the performance of the three forwarding policies when the neighborhood table of each node contains all forwarding choices. The link quality estimators are initialized according to the USC link model. This set of experiments is designed to quantify the best-case performance of the forwarding policies in the presence of perfect knowledge about the neighborhood and link qualities. Next, we consider the case when the neighborhood table has limited size and the link quality of each forwarding choice is estimated online. Finally, we evaluate the impact of different workloads on RPAR.

A. Performance of Forwarding Policy

The first set of experiments uses a light workload generated by three sources. Each source sends on average a packet every 4 s. To evaluate RPAR's ability to adapt to different real-time delay requirements, we vary the packet deadline between 100 ms and 350 ms. Figure 2(a) shows the miss ratio as the deadline is varied. The forwarding policies that use the default transmission power, MinE_L and MaxV_L, start missing deadlines when the deadline is 350 ms. As the deadline decreases, they miss an increasing number of deadlines until 200 ms when none of the transmitted packets meet their deadline. In contrast, the baselines using the maximum transmission power, MinE_H and MaxV_H, have significantly lower miss ratios. This result confirms our observation (see Section II) that using a high transmission power can effectively reduce communication delay.

However, Figure 2(b) shows that the baseline protocols using high transmission power consume significantly more energy per packet. In contrast, RPAR consistently achieves both desired real-time performance and energy efficiency under different deadlines. As shown in Figure 2(a), RPAR achieves miss ratios close to MinE_H and MaxV_H. At the same time, as shown in Figure 2(b), RPAR consumes less energy than MinE_L and MaxV_L for all deadlines except 100ms. This is because our forwarding policy selects more energy-efficient forwarding choices that still meet the delay requirements. Note the correlation between the energy consumption and the deadline: RPAR spends additional energy to meet tighter deadlines. This shows the desired trade-off between real-time performance and energy efficiency.

B. Performance with Neighborhood Management

This set of experiments is designed to evaluate the performance of the forwarding policies when using neighborhood management. In the following experiments, RPAR uses the

neighbor discovery scheme described in Section IV-D. In our simulations, we tune α so that it takes four iterations to increase the power from the default power level to the maximum power. We set $\beta = 1$. Similar to the MT protocol [7], the baselines use a neighborhood manager that uses beacons for neighborhood discovery and the FREQUENCY algorithm for table management. In all experiments, each node sends beacons with a period of 20 s using the same transmission power as the data packets. When the periodic beacon scheme is used, data packets start to be transmitted after 40 s to allow the link quality estimators in the neighborhood table to be initialized. The size of the neighbor table is set to 360 bytes for all protocols.

The performance of RPAR is affected by the quality of the forwarding choices found in the neighborhood table. As such, we consider three versions of RPAR. RPAR$_{best}$ quantifies the performance of the forwarding policy when the table is pre-filled, representing the best-case performance of RPAR. RPAR$_{cold}$ starts with an empty table and builds its neighborhood table according to the neighborhood management scheme described in Section IV-D. Since in practice the neighborhood table is usually not empty, RPAR$_{cold}$ represents the worst-case performance of RPAR. Therefore, we introduce RPAR$_{warm}$, which approximates the average-case performance when some forwarding choices are already in the table after routing the first 50 packets.

Figure 3 shows the performance of the forwarding policies used in combination with their respective neighborhood management policies. Figure 3(a) indicates that miss ratios of all protocols increased due to imperfect knowledge about forwarding choices. We observe a significant performance degradation (in terms miss ratio and energy consumption) when the beacon-based neighborhood manager is used with the baseline protocols whereas our on-demand neighborhood manager has a small impact on RPAR's performance. In contrast to the previous set of experiments where the baselines using the maximum transmission power had miss ratios comparable to those attained by RPAR, in these experiments RPAR clearly outperforms them. This shows that neighborhood management is an important issue in power-aware routing. The benefit of the new neighborhood manager is particularly evident for tight deadlines. At 150ms, the miss ratios of RPAR$_{cold}$ and RPAR$_{warm}$ are *only* 4.7% and 1.2% higher than RPAR$_{best}$, respectively. In contrast the miss ratio of MaxV$_H$ jumps up by 30.5%. Two factors contributed to the improved performance of RPAR's neighborhood discovery over the beacon based scheme. First, our neighborhood manager is deadline-aware in that it discovers and keeps forwarding choices that satisfy the velocity requirement in the neighborhood table. Furthermore, our power adaptation and neighbor discovery schemes find good forwarding choices faster than the periodic beacons.

Figure 3(b) shows the energy consumed per data packet, including the energy spent for transmitting the overhead packets. Figure 3(c) shows the energy consumed for transmitting *only* the overhead packets. Figures 3(b) and 3(c) indicate that the energy consumed by the baseline protocols for neighborhood discovery accounts for a large part of the energy consumed per data packet. In contrast, RPAR which uses the on-demand neighborhood discovery scheme consumes significantly less energy. The reduction in energy consumption is attributed to both our forwarding policy (see Figure 2(b)) and our neighborhood manager which introduces significantly lower overhead. While the beacon period may be increased to lower the energy consumption, this would further degrade the real-time performance of the baselines.

C. Impact of Workload

The final set of experiments evaluates the performance of RPAR under different workloads. Figures 4(a) and 4(b) show the experimental results for the case when the workload is varied by changing the number of sources from 4 to 10. Each source generates data with an inter-packet time of 6 s. The deadline is fixed at 300 ms. We observe that the number of deadline misses increases with the number of sources for all protocols except MinE$_L$. Because of the large confidence intervals of MinE$_L$, no clear correlation between its miss ratio and the increase in the number of sources may be established. The wide confidence intervals are the result of MinE$_L$ selecting unreliable links for transmission. The increase in the miss ratio with the number of source of the other protocols may be attributed to higher contention. The RPAR protocols have lower miss ratios and higher energy efficiency than the baselines.

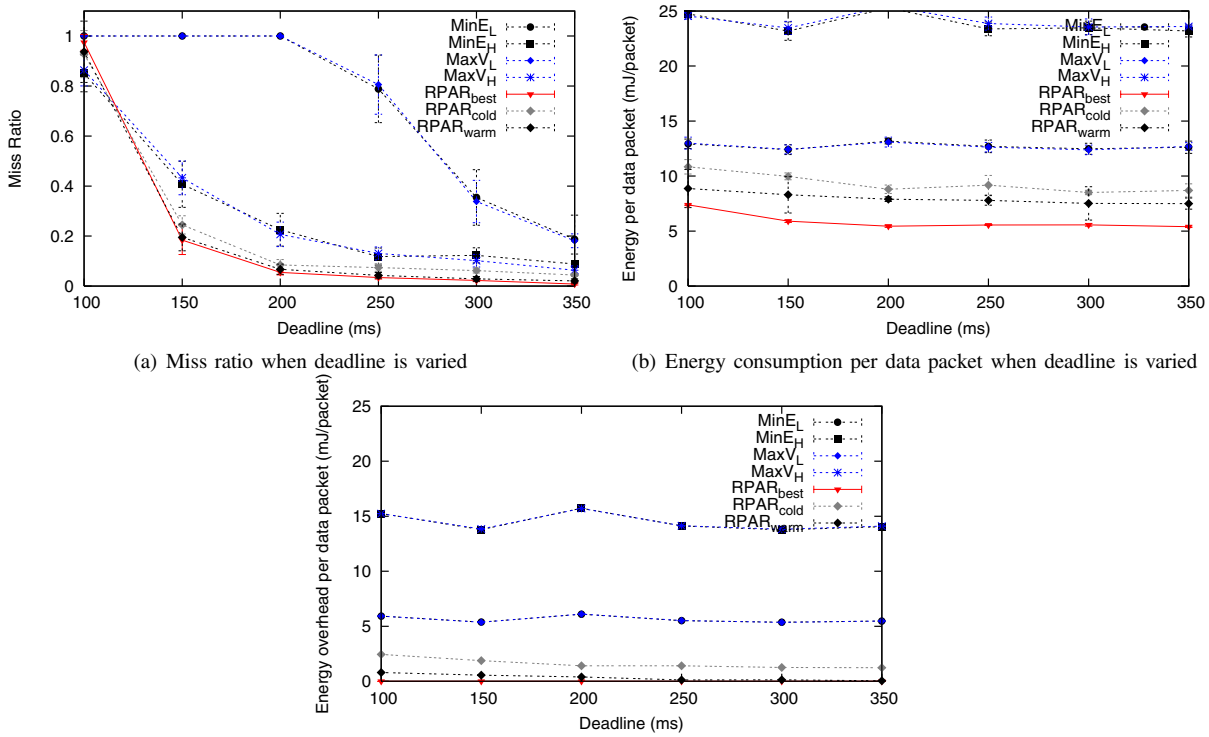
Figures 4(c) and 4(d) show the experimental results for the case when the average inter-packet time of each source is varied from 1s to 5s and the number of sources is fixed at 3. As the inter-arrival time is increased, the deadline miss ratios decrease for all protocols due to reduced network load. Similar to previous experiments, RPAR's miss ratio is lower than those of the baselines in all tested settings. Figure 4(d) indicates a increase in the total energy consumed by the baselines and a decrease in energy per data packet for RPAR. The increase in energy per data packet for the baselines is attributed to a lower number of dropped packets as the inter-packet packet time increases. RPAR incurs slight decrease in energy per data packet because it adaptively lowers transmission power under lower network loads. Overall RPAR consistently outperforms the baselines in term of energy efficiency when the inter-arrival time is 1 s.

VI. Discussion

We now identify several open issues that have not been addressed in our current work and discuss how RPAR can be extended to address them.

A. Handling Congestion

RPAR's power-adaptation policy exhibits a pathological behavior when a node is congested. Due to high contention, a node retransmits a packet several times before it is successfully received. Hence, RPAR increases the transmission power, which may further increase contention. RPAR can be integrated with existing protocols to deal with this problem. At the MAC layer, several methods have been proposed to



(a) Miss ratio when deadline is varied

(b) Energy consumption per data packet when deadline is varied

(c) Overhead energy consumption per data packet when deadline is varied

Fig. 3. Performance of considered protocols when deadline is varied (with neighborhood management).

differentiate between packets lost due to collisions or due to poor link quality [18]. Such feedback from the MAC layer would prevent RPAR from needlessly increasing the transmission power under congestion. Alternatively, RPAR may also work with existing congestion detection mechanisms for WSNs [19][20]. When a node detects congestion, RPAR should stop increasing the transmission power to avoid worsening the congestion and to allow the congestion control protocols [19][20][21] to alleviate it. Furthermore, RPAR may also be integrated with congestion and rate control techniques to keep the network below its real-time capacity bound [9].

B. Handling Holes

A known problem with greedy geographic forwarding is that it may fail to find a route in the presence of holes in the network topology. Such holes may appear due to voids in node deployment or node failures. RPAR partly mitigates this problem through power control: if the diameter of the hole is smaller than the transmission range at the maximum power, RPAR can transmit packets across the hole. For networks with large holes, RPAR needs to incorporate face routing mechanisms [22][23][24][25][26] to route packets around them.

When there are large holes, the Euclidean distance becomes a poor approximation of the actual path length. As a result, RPAR computes the expected number of hops incorrectly. We alleviate this problem through the dynamic velocity assignment policy, which recomputes the required velocity based on the progress toward the destination. The performance of RPAR may be further improved by considering the dilation of a path. This may be estimated by computing the boundary of a hole using a protocol such as BoundHole [26].

C. Integration with Power Management

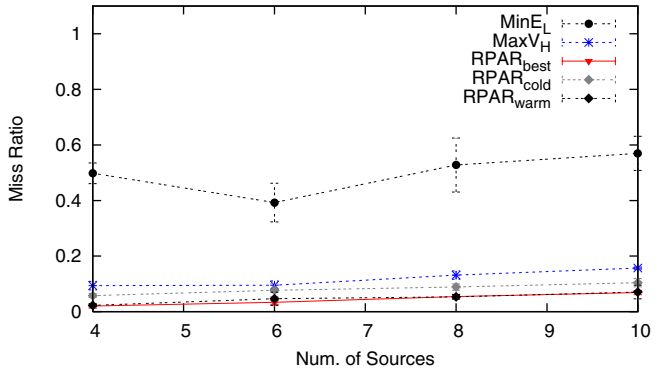
RPAR aims to minimize the energy consumed for packet transmission. However, the cost of packet transmissions is only a part of the total energy consumption of a network. To further reduce total energy consumption, a WSN needs to integrate RPAR with a power-management protocol that reduces the energy wasted on idle listening. We consider two classes of power-management techniques and describe how RPAR may be integrated with them.

An effective power-management approach is to maintain a connected backbone composed of nodes that are always active, while the other nodes typically follow a periodic sleep schedule to save energy (e.g., [27][28]). The backbone is used for routing and buffering packets destined to sleeping nodes. The last-hop delay to a sleeping node is usually bounded by the period of its duty cycle and can be accounted for by adjusting the packet's velocity requirement.

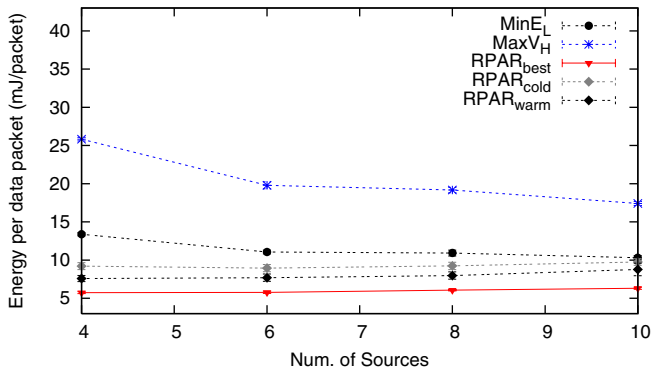
Sleep scheduling algorithms alternate periods of sleep and activity. Of particular interest to real-time applications are sleep scheduling algorithms that adjust their periods of sleep and activity based on observed workload to minimize the impact of sleep schedules on message delay, such as T-MAC [29], 802.11 Power Saving Mode, ESSAT [30], on-demand power management [31], and the low-power listening scheme adopted by B-MAC [5]. As the packet is routed towards the destination, RPAR's dynamic deadline assignment policy can account for the additional delay introduced by sleep scheduling.

VII. Related Work

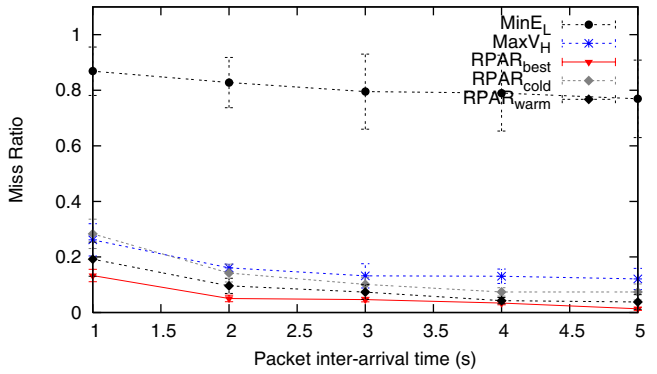
RPAR is related to LAPC, SPEED, and MM-SPEED. LAPC [32] is a power-control protocol designed to reduce commu-



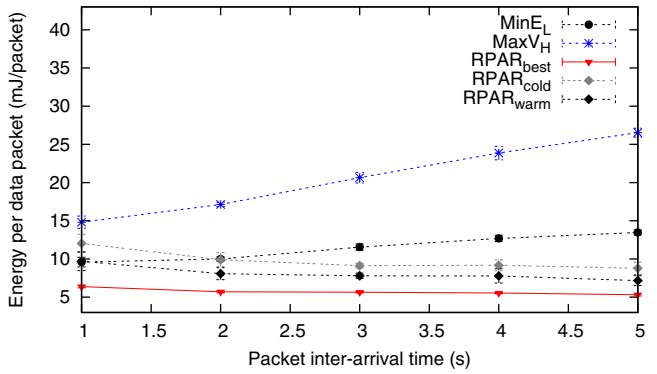
(a) Miss ratio when number of sources is varied



(b) Energy consumption per data packet when number of sources is varied



(c) Miss ratio when data rate is varied



(d) Energy consumption per data packet when data rate is varied

Fig. 4. Performance of considered protocols when the workload is varied (with neighborhood management).

nication delays by adapting the transmission power to the workload. LAPC is not concerned with packet deadlines and only reduces communication delays in a best effort fashion. In contrast, SPEED, MM-SPEED, and RPAR are designed for real-time applications with explicit delay requirements. SPEED [12] bounds the end-to-end communication delay by enforcing a uniform delivery velocity (called *speed* in [12] and [33]). MM-SPEED [33] extends SPEED to support different delivery velocities and levels of reliability. Both SPEED and MM-SPEED use fixed transmission power. In addition, RPAR differs from the above protocols in the following important respects. First, RPAR is the only protocol that integrates power control and real-time routing for supporting energy-efficient real-time communication. Furthermore, it allows the application to control the trade-off between energy consumption and communication delay by specifying packet deadlines. Second, unlike the other protocols, RPAR is designed to handle lossy links. This is an important feature since lossy links are common in WSNs and have a profound effect on communication delay [3][34]. Third, RPAR employs a novel neighborhood management mechanism that is more efficient than the periodic beacons scheme adopted by LAPC, SPEED and MM-SPEED. The simulations indicate that neighbor management has a significant impact on both real-time performance and energy efficiency.

There has been significant research on quality of service (QoS) support in wireless ad hoc networks. Several mechanisms to provide QoS support in 802.11 have been proposed. The most common approach is to provide service differentiation [35][36][37][38][39] by manipulating different MAC parameters. Overviews of these approaches are presented in [40] and [41]. Another approach for achieving QoS is to provide statistical guarantees on real-time traffic through online admission and rate control [21][42][43]. MAC layer prioritization, admission control and rate control may be used in conjunction with RPAR to further improve its real-time performance. Other papers propose routing protocols that provide QoS through path discovery and resource reservation [44][45] but none of the them use power control to achieve desired QoS.

Power-aware routing has been investigated in several previous works. For example, Singh et al. propose five power-based routing metrics that can be used to minimize power consumption or extend system lifetime [46]. Several power-aware protocols have been proposed to maximize network lifetime [47][48][49][48]. Power-aware routing has been implemented on real wireless network platforms [50]. Gomez and Campbell [51] provide theoretical analysis showing that allowing each node to dynamically adjusting its transmission power leads to improved capacity and energy-efficiency over the case when all nodes use a common transmission power. Unlike RPAR, none of the above power-aware routing protocols is designed to support real-time communication.

VIII. Conclusions

We have developed RPAR, the first real-time power-aware routing protocol for WSNs. In contrast to existing protocols that treat real-time performance and energy efficiency in isolation, RPAR integrates novel real-time routing and dynamic power adaptation algorithms to achieve application-specified communication delays at low energy cost. Another distinguishing feature of RPAR is that it handles realistic properties of WSNs such as lossy links, limited memory, and bandwidth. Simulations based on a realistic radio model of MICA2 motes show that RPAR significantly reduces the deadline miss ratio and energy consumption compared with existing real-time and energy-efficient routing protocols.

IX. Acknowledgement

This work is funded in part by NSF under ITR grants CCR-0325529 and CCR-0325197.

References

[1] T. He, P. A. Vicaire, T. Yan, L. Luo, L. Gu, G. Zhou, R. Stoleru, Q. Cao, J. A. Stankovic, and T. Abdelzaher, "Achieving real-time target tracking using wireless sensor networks." in *To appear in RTAS'06*.

[2] C. Lu, G. Xing, O. Chipara, C.-L. Fok, and S. Bhattacharya, "A spatiotemporal query service for mobile users in sensor networks," in *ICDCS '05*, 2005.

[3] J. Zhao and R. Govindan, "Understanding packet delivery performance in dense wireless sensor networks," in *SenSys '03*, 2003, pp. 1–13.

[4] A. Cerpa, J. Wong, L. Kuang, M. Potkonjak, and D. Estrin, "Statistical model of lossy links in wireless sensor networks," in *IPSN '05*, 2005.

[5] J. Polastre, J. Hill, and D. Culler, "Versatile low power media access for wireless sensor networks," in *SenSys '04*, 2004.

[6] M. Zuniga and B. Krishnamachari, "Analyzing the transitional region in low power wireless links," in *SECON '04*, 2004.

[7] A. Woo, T. Tong, and D. Culler, "Taming the underlying challenges of reliable multihop routing in sensor networks," in *SenSys '03*, 2003.

[8] P. Gupta and P. R. Kumar, "The capacity of wireless networks," *IEEE Trans on Information Theory*, vol. 46, no. 2, 2000.

[9] T. F. Abdelzaher, S. Prabh, and R. Kiran, "On real-time capacity limits of multihop wireless sensor networks," in *RTSS '04*, Dec. 2004.

[10] J. Hightower and G. Borriello, "Location systems for ubiquitous computing," *IEEE Computer*, 2001.

[11] K. Seada, M. Zuniga, A. Helmy, and B. Krishnamachari, "Energy efficient forwarding strategies for geographic routing in wireless sensor networks," in *SenSys '04*, 2004.

[12] T. He, J. Stankovic, C. Lu, and T. Abdelzaher, "Speed: A stateless protocol for real-time communication in sensor networks," in *ICDCS '03*, 2003.

[13] V. Jacobson, "Congestion avoidance and control," in *SIGCOMM '88*.

[14] E. D. Demaine, A. Lopez-Ortiz, and J. I. Munro, "Frequency estimation of internet packet streams with limited space," in *Proceedings of the 10th Annual European Symposium on Algorithms*, 2002, pp. 348–360.

[15] G. Simon, P. Volgyesi, M. Maroti, and A. Ledeczi, "Simulation-based optimization of communication protocols for large-scale wireless sensor networks," in *IEEE Aerospace Conference '03*, 2003.

[16] "Crossbow, inc., mica2 mote," http://www.xbow.com/Products/productsdetails.aspx?sid=72, Apr. 19 2006.

[17] G. Zhou, T. He, S. Krishnamurthy, and J. A. Stankovic, "Impact of radio irregularity on wireless sensor networks," in *MobiSYS '04*, 2004.

[18] A. Woo, K. Whitehouse, F. Jiang, J. Polastre, and D. Culler, "The shadowing phenomenon: implications of receiving during a collision," in *UCB Technical Report*, 2004.

[19] B. Hull, K. Jamieson, and H. Balakrishnan, "Mitigating congestion in wireless sensor networks," in *SenSys '04*.

[20] C.-Y. Wan, S. B. Eisenman, and A. T. Campbell, "Coda: congestion detection and avoidance in sensor networks," in *SenSys '03*.

[21] G.-S. Ahn, A. Campbell, A. Veres, and L.-H. Sun, "Swan: Service differentiation in stateless wireless ad hoc networks," in *INFOCOM '02*.

[22] B. Karp and H. T. Kung, "Gpsr: greedy perimeter stateless routing for wireless networks," in *MOBICOM '00*, 2000, pp. 243–254.

[23] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia, "Routing with guaranteed delivery in ad hoc wireless networks," *Wireless Networks*, vol. 7, no. 6, 2001.

[24] F. Kuhn, R. Wattenhofer, Y. Zhang, and A. Zollinger, "Geometric ad-hoc routing: of theory and practice," in *PODC '03*, 2003, pp. 63–72.

[25] Y.-J. Kim, R. Govindan, B. Karp, and S. Shenker, "Geographic routing made practical," in *USENIX '05*.

[26] Q. Fang, J. Gao, and L. Guibas, "Locating and bypassing routing holes in sensor networks," in *INFOCOM '04*, 2004.

[27] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris, "Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks," in *MOBICOM '01*, 2001, pp. 85–96.

[28] Y. Xu, J. Heidemann, and D. Estrin, "Geography-informed energy conservation for ad hoc routing," in *MOBICOM '01*, 2001, pp. 70–84.

[29] T. van Dam and K. Langendoen, "An adaptive energy-efficient mac protocol for wireless sensor networks," in *SenSys '03*, 2003.

[30] O. Chipara, C. Lu, and G.-C. Roman, "Efficient power management based on application timing semantics for wireless sensor networks," in *ICDCS '05*.

[31] R. Zheng and R. Kravets, "On-demand power management for ad hoc networks," in *INFOCOM '03*, 2003.

[32] M. R. Fouad, S. Fahmy, and G. Pandurangan, "Latency-sensitive power control for wireless ad-hoc networks," in *Q2SWinet '05*, 2005.

[33] E. Felemban, C.-G. Lee, E. Ekici, R. Boder, and S. Vural, "Probabilistic qos guarantee in reliability and timeliness domains in wireless sensor networks," in *INFOCOM '05*, 2005.

[34] D. Son, B. Krishnamachari, and J. Heidemann, "Experimental study of the effects of transmission power control and blacklisting in wireless sensor networks," in *SECON '04*, 2004.

[35] Y. Ge and J. Hou, "An analytical model for service differentiation in ieee 802.11," in *ICC '03*, 2003.

[36] I. Aad and C. Castelluccia, "Differentiation mechanisms for IEEE 802.11," in *INFOCOM '01*, 2001, pp. 209–218.

[37] V. Kanodia, C. Li, A. Sabharwal, B. Sadeghi, and E. Knightly, "Distributed multi-hop scheduling and medium access with delay and throughput constraints," in *MobiCom '01*, 2001.

[38] IEEE, "Draft supplement to ieee standard for telecommunications and information exchange between systems lanman specific requirements,. part 11: Wireless lan medium access control (mac) and physical layer (phy)," *IEEE Standard 802.11*, Oct 2002.

[39] S. Mangold, S. Choi, P. May, O. Klein, G. Hiertz, and L. Stibor, "Ieee 802.11e wireless lan for quality of service," in *Proceedings of the European Wireless*.

[40] H. Zhu, M. Li, I. Chlamtac, and B. Prabhakaran, "A survey of quality of service in ieee 802.11 networks," in *IEEE Wireless Communications*, 2004.

[41] W. Pattara-Atikom, P. Krishnamurthy, and S. Banerjee, "Distributed mechanisms for quality of service in wireless lans," *IEEE Wireless Communications*, 2003.

[42] M. G. Barry, A. T. Campbell, and A. Veres, "Distributed control algorithms for service differentiation in wireless packet networks," in *INFOCOM '01*, 2001, pp. 582–590.

[43] Y. Yang and R. Kravets, "Distributed qos guarantees for realtime traffic in ad hoc networks," in *SECON '04*, 2004.

[44] S. Chen and K. Nahrstedt, "A distributed quality-of-service routing in ad-hoc networks," *IEEE Journal on Selected Areas in Communications*, 1999.

[45] P. Sinha, R. Sivakumar, and V. Bharghavan, "CEDAR: a core-extraction distributed ad hoc routing algorithm," in *INFOCOM '99*, 1999.

[46] S. Singh, M. Woo, and C. S. Raghavendra, "Power-aware routing in mobile ad hoc networks," in *MOBICOM '98*, 1998.

[47] Q. Li, J. Aslam, and D. Rus, "Online power-aware routing in wireless ad-hoc networks," in *MOBICOM '01*, 2001.

[48] J.-H. Chang and L. Tassiulas, "Energy conserving routing in wireless ad-hoc networks," in *INFOCOM '00*, 2000.

[49] A. Sankar and Z. Liu, "Maximum lifetime routing in wireless ad-hoc networks," in *INFOCOM '04*, 2004.

[50] S. Doshi, S. Bhandare, and T. X. Brown, "An on-demand minimum energy routing protocol for a wireless ad hoc network," *SIGMOBILE '02*, no. 3, 2002.

[51] J. Gomez and A. Campbell, "A case for variable-range transmission power control in wireless multihop networks," in *INFOCOM '04*.