Practical Lazy Scheduling in Sensor Networks

Ramana Rao Kompella and Alex C. Snoeren
University of California, San Diego
{ramana,snoeren}@cs.ucsd.edu

ABSTRACT

Experience has shown that the power consumption of sensors and other wireless computational devices is often dominated by their communication patterns. We present a practical realization of lazy packet scheduling that attempts to minimize the total transmission energy in a broadcast network by dynamically adjusting each node's transmission power and rate on a per-packet basis. Lazy packet scheduling leverages the fact that many channel coding schemes are more efficient at lower transmission rates; that is, the energy required to send a fixed amount of data can be reduced by transmitting the data at a lower bit rate and transmission power.

The optimal per-packet transmission rate in a multi-node network is governed in practice by the available bit rates of the given transceiver(s), the nodes' delay tolerance, and the offered load at every node contending for the shared broadcast channel. We propose an extension to the traditional CSMA/CA MAC scheme called L-CSMA/CA that allows individual nodes to continually estimate the current demand for a broadcast channel and adjust their transmission schedules accordingly. Our simulation results show that L-CSMA/CA can provide improved energy efficiency in a single-hop, broadcast network (20–25% with more than 10 nodes, and up to 99% for four nodes with a standard power function) for both Poisson and bursty arrivals with only minor degradation the capacity of the channel.

1. INTRODUCTION

Energy is often the most precious resource in networks of sensors and other battery-powered wireless devices. In many of these devices, the communication component is the largest contributor to power consumption. This paper explores power-saving techniques based on the observation that, in many cases, the sustained bandwidth utilization of the network is significantly less than the maximum useful channel capacity. Furthermore, many channel coding schemes are more power-efficient at lower transmission rates; that is, the energy required to send a packet can be reduced by transmitting the packet at a lower bit rate and transmission power. Hence, significant power savings may be realized by transmitting data as slow as possible, often termed *lazy* packet scheduling.

SenSys'03, November 5–7, 2003, Los Angeles, California, USA.

Unfortunately, packets cannot be transmitted arbitrarily slowly on a shared channel. If each node in a broadcast network reduces its transmission speed without regard for the demand at other nodes the resulting decrease in channel utilization could impact overall network throughput. The optimal per-packet transmission rate in a broadcast network is governed in practice by a large number of factors including the available bit rates of the radio devices, the average per-packet delay at each node, and the load at every node contending for the channel.

Commodity radios typically have a relatively small, fixed set of transmission rates from which to choose. The round-off errors introduced by attempting to use a transmission schedule computed assuming a continuously variable bandwidth radio often eliminate most of the potential energy savings. Starting from a known, optimal offline algorithm for lazy packet scheduling [28], we develop an online algorithm that considers a given set of available, discrete transmission rates. By analyzing the performance of our algorithm over a range of available transmission speeds, we identify critical design points that make certain multi-rate radios particularly amenable to lazy scheduling.

Current sensors and embedded network devices also tend to have limited processing power. For example, the Berkeley MICA2 Mote, now commercially available from Crossbow [6], uses a 4 MHz processor with 128 kilobytes of Flash memory. Such limited computational capacity makes it impractical to consider complicated packet scheduling algorithms. Furthermore, the CPU power spent calculating the packet transmission schedule itself can approach the possible energy savings if the algorithm is too complicated. Hence, we present an extremely simple and easy to implement online algorithm that compares favorably to the optimal offline algorithm.

Of course, from the point of view of any individual node, an optimal transmission schedule will transmit packets almost constantly (assuming the radio is capable of a sufficiently low bit rate). For networks made up of multiple nodes sharing a broadcast channel, this schedule may not be optimal for the network as a whole. Instead, the optimal transmission schedule has each packet sent as slowly as possible subject to the constraint that all nodes are able to transmit their packets. We present a practical, distributed algorithm for CSMA/CA broadcast radio technologies (as used in 802.11 [10] networks, for example) called L-CSMA/CA that allows each node in a broadcast, single-hop, many-to-many network to independently adjust its transmission power and rate to conserve energy without significantly impacting the capacity of the channel nor each node's ability to contend for an over-subscribed channel. We do not, however, consider the interference issues raised in multi-hop networks for the purposes of this paper.

Our simulation results show that our offline algorithm for discretized bandwidths is a good approximation of the continuous of-

fline algorithm, and that our online version is able to achieve significant transmission power savings for traffic with both Poisson and bursty arrival patterns while meeting arbitrary packet deadlines. The effectiveness of our online algorithm scales with the delay tolerance of the network and achieves significant transmission energy savings even with low delay tolerances. Finally, we show that L-CSMA/CA is an effective approximation of a distributed online scheduling algorithm.

The power-saving effectiveness of L-CSMA/CA degrades gracefully with increasing number of nodes and has only minor impact on the channel capacity across the entire range of offered loads. For a representative transmission power function [28], L-CSMA/CA can achieve an average transmission power savings in simulation of 20–25% with more than 10 nodes and up to a 99% compared to CSMA/CA in a 20%-loaded, four-node network. We do not claim, however, that all tranceivers are implemented in a way that enables them to benefit from reduced transmission energy requirements—the power savings achievable by a particular device depends on a number of factors including the set of available bitrates, the mechanisms used to adjust transmission power and rate, and various radio inefficiencies.

The remainder of this paper is organized as follows. We begin in Section 2 with a brief overview of previous work in lazy packet scheduling and alternate approaches. Section 3 demonstrates how existing theoretical results on optimal offline lazy scheduling algorithms for continuously variable transmission rates can be adapted to discrete bandwidth levels. We present our novel, discretized online algorithm in Section 4, which we use as a basis for the implementation of L-CSMA/CA in Section 5. We discuss the practicality and limitations of our scheme in Section 6 before suggesting areas for future work in Section 7.

2. BACKGROUND AND RELATED WORK

Information theory tells us that for any channel, there exists a code of rate R for all rates up to the channel capacity (often termed the Shannon limit) [5]. Furthermore, it is known that, using an optimum set of encoding schemes, the energy needed to transmit a fixed amount of information is convex with respect to the rate of the code [1]. That is, the energy required to send a packet can be reduced by transmitting the packet with a lower bit rate encoding. Hence, an energy-conserving transmitter should attempt to transmit packets at the slowest possible rate. This technique, known as lazy packet scheduling, was analyzed in the case of a private channel by Uysal-Biyikoglu, Prabhakar, and El Gamal [28]. They initially studied the theoretical case of a single sender with a near-optimal transmission device with continuously variable speed and power settings.

Unfortunately, commodity radios typically provide only a small, fixed set of transmission powers and encoding schemes to choose from. Schurgers, Raghunathan, and Srivastava were the first to consider adjusting the modulation scheme employed by a transmitter to implement lazy packet scheduling [25]. By adjusting the constellation size of Quadrature Amplitude Modulation (QAM)—a technique they called dynamic modulation scaling (DMS)—Schurgers *et al.* showed that well-defined real-time flows can be effectively scheduled using an admission control mechanism resulting in significant power savings. In our work we do not require that sources define flows or specify their bandwidth utilization *a priori*, but observe that QAM with variable consetellation sizes may be an effective method of implementing variable transmission speeds. Raghunathan, Ganeriwal, Schurgers, and Srivastava also presented an extended weighted fair queuing scheme (E^2WFQ) that dynamically adjusts the level of output buffering used in DMS to trade off power savings against packet delay [21]. E^2WFQ might be useful to dynamically adjust the look-ahead buffer used in L-CSMA/CA; we have not yet explored this synergy.

Researchers have previously proposed methods of determining the most efficient transmission power level for each node in a wireless network while maintaining connectivity [22]. Because nodes may not be evenly distributed in a wireless network, the transmission power required for a given bitrate may differ from destination to destination [12]. Uysal-Biyikoglu *et al.* extended their lazy scheduling approach to consider a single transmitter with multiple receivers with possibly different power functions [7]. Uysal-Biyikoglu and El Gamal recently showed that their MoveRight algorithm developed for the single sender, multiple receiver model can be adapted to the multiple sender, single receiver case [27]. Unfortunately, the resulting FlowRight algorithm can be quite computationally expensive, requiring a large number of iterations to compute a packet schedule.

Recently, several researchers have begun considering more sophisticated scheduling constraints. Schurgers and Srivastava, for example, observed that the transmission power required for a given rate may also vary over time due to interference, fading, and other channel characteristics. They extended their DMS technique to opportunistically vary transmission speeds based upon instantaneous channel conditions in order to meet packet deadlines in the most energy efficient fashion [26]. Harnessing the regenerative nature of modern batteries, Nuggehalli, Srinivasan, and Rao showed that battery utilization can be improved by inserting additional recovery periods into lazy transmission schedules [17].

With the sole exception of Uysal-Biyikoglu and El Gamal's complex FlowRight algorithm, which assumes strict coordination between senders, all of these schemes consider lazy packet scheduling for a private channel. To our knowledge, we are the first to propose and evaluate a simple, distributed MAC protocol for multi-node, single-hop broadcast networks that automatically adjusts both transmission rate and power in response to excess capacity on a shared channel.

Lazy packet scheduling is specific instance of a more general area known as dynamic speed scaling. A well-studied topic, dynamic speed scaling considers adjusting the speed of a processor to minimize total energy while ensuring that each job is completed before its deadline. This problem is especially important in the embedded systems environment, where processor speeds are typically adjusting through dynamic voltage or frequency scaling [16, 20, 23, 29]. In the standard formulation of the dynamic speed scaling problem, however, jobs are preemtable; packet transmission, on the other hand, does not enjoy that luxury. Our offline and online algorithms turn out to be special instances of more general speed scaling approximation algorithms with known performance bounds [11, 30].

Of course, lazy packet scheduling is not the only approach to saving communication power. For radios with non-convex transmission curves, such as those that consume significant power while idle, one simple approach is to turn the radio off or shift it into a low-power sleep mode when not in use. The down side is that a node cannot receive data in sleep mode. Hence, effective use of sleep mode requires a significant degree of coordination between the nodes. Despite this requirement, approaches that turn off radios for some period of time have been well-studied in the literature. The 802.11 specification includes a power-saving ad-hoc mode [10], but it is known to have a significant impact on transmission throughput. Other approaches have been shown to be more efficient. For instance, Span considers turning off unneeded relay nodes in a multiply connected multi-hop network [2].

Significant effort has been spent measuring the power consumption of various link and network technologies. Feeney and Nilsson conducted one of the earliest studies of the power consumption of actual 802.11 devices [9]. Chen *et al.* analytically compared the power consumption of various MAC protocols and proposed a new, more energy-efficient protocol [3]. Researchers have achieved additional power savings by utilizing application-level information in network- and link-layer scheduling decisions. This is especially useful when trying to maintain a notion of "liveness" in the network, such as keeping each node reachable for the longest possible time. As just one example, SPIN uses application knowledge to efficiently disseminate sensor data using lower total network energy [15]. Price *et al.* achieved a similar goal by granting nodes with lower power reserves priority access to a wireless medium [19].

3. OFFLINE ALGORITHMS

An optimal offline lazy packet scheduling algorithm for one node with a continuously variable radio transmitting over a private channel was given by Uysal-Biyikoglu *et al.* [28]. Intuitively, the algorithm attempts to divide the available transmission time evenly among all the packets subject to constraints imposed by the packet arrival times and any packet deadlines. By 'arrival time,' we mean the time at which a packet is submitted to the network layer at a node for transmission over the wireless channel. In this section we examine the applicability of the continuous algorithm to the discrete-bandwidth case and present an extended algorithm that provides a reasonable approximation to the continuous case.

3.1 Optimal offline algorithm

An optimal offline algorithm, by definition, knows the set of packet arrivals and tries to schedule the packets as energy-efficiently as possible within a given interval. We briefly introduce some notation before presenting an algorithm to identify an efficient transmission schedule.

Let us assume for simplicity that all packets are of equal length; it is straightforward to extend the algorithm to consider variable-length packets. Define the transmission energy $\omega(\tau)$ of a packet with transmission duration τ as the amount of energy necessary to send the packet over time τ. Recall that we assume the energy function is strictly convex in transmission duration, so $\omega(\tau)$ decreases with increasing τ; we will examine the factors governing the convexity of $\omega(\tau)$ in Section 6. Suppose that the inter-arrival times $d_1, d_2, \ldots, d_M$ for the M packets that arrive in the interval $[0, T)$ are known in advance, i.e., before *t=0*. (We can assume, without loss of generality, that packet 0 arrives at time 0.) The offline scheduling problem is then to determine the transmission duration vector $\vec{\tau}$ so as to minimize $\omega(\vec{\tau}) = \sum_{i=1}^{M} \omega(\tau_i)$.

Let $k_0 = 0$. Define

$$m_1 = \max_{k \in 1,\ldots,M} \left\{ \frac{1}{k} \sum_{i=1}^{k} d_i \right\}$$

and

$$k_1 = \max \left\{ k : \frac{1}{k} \sum_{i=1}^{k} d_i = m_1 \right\}.$$

For $j \geq 1$, let

$$m_{j+1} = \max_{k \in \{1,\ldots,M\}} \left\{ \frac{1}{k} \sum_{i=1}^{k} d_{k_j+i} \right\} \quad (1)$$

and

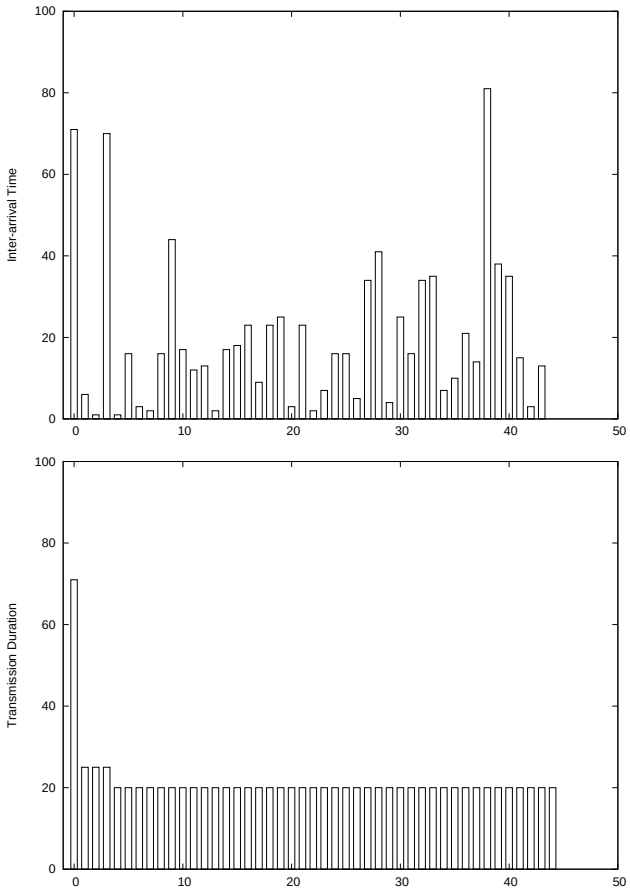

Figure 1: The inter-arrival times, d_i for 45 packets over a particular interval $[0, T)$ and their respective transmission durations, τ_i calculated by the offline algorithm.

$$k_{j+1} = k_j + \max \left\{ k : \frac{\sum_{i=1}^{k} d_{k_j+i}}{k} = m_{j+1} \right\}. \quad (2)$$

where *k* varies between 1 and $M - k_j$. These pairs (m_j, k_j) are used to obtain the schedule $\vec{\tau}$ defined as

$$\tau_i = m_j \text{ if } k_{j-1} < i \leq k_j.$$

$\vec{\tau}$ has been shown to be optimal [28]; we do not repeat the proof here. Therefore, $\vec{\tau}$ gives us a lower bound on the energy consumption for all our later comparisons and calculations. An example of the above algorithm is shown in Figure 1. The top graph shows the inter-arrival periods d_i for each of 45 packets that arrived in the interval $[0, T)$. The bottom graph shows the resulting transmission schedule, $\vec{\tau} = [\tau_0, \ldots, \tau_{44}]$, where the time interval has been spread out across the individual packet transmission times as evenly as possible given the packet arrival times.

3.2 Discretized offline algorithm

One of the implicit assumptions in the original proof of optimality for the algorithm is that bandwidth could be continuously varied. While true in theory, practical radios are generally only able to transmit at a small set of discrete bandwidth levels, and, therefore, cannot continuously vary packet transmission durations. While the offline schedules computed by the algorithm above can be adjusted

to take into account the set of available, discrete bandwidth levels, there are interesting differences in the energy consumed per packet or the average delay suffered by packets depending on discretization method selected.

First, we define two straightforward, natural ways to discretize the offline algorithm. The first approach rounds the transmission duration calculated by the optimal offline algorithm up to the nearest value greater than it (ceiling) corresponding to an available bandwidth level. If there is no such value available, then the maximum transmission duration (corresponding to the lowest available bandwidth) is chosen. Conversely, the second approach rounds the transmission duration down to the nearest lower value. If there are no such values available, the minimum transmission duration (highest bandwidth) is selected.

Let $b_1, b_2, \ldots, b_B$ be the set of B available bandwidths that the radio is capable of transmitting, in descending order. If p is the size of a packet, then the possible transmission durations are denoted by t_i, $i \in 1, \ldots, B$ and $t_i = \frac{p}{b_i}$. Since the bandwidths are in descending order, the transmission durations are in ascending order. Let τ_i be the transmission duration assigned by the optimal offline algorithm described in the previous section. We now formally state the two approaches to discretize the optimal offline algorithm.

Ceiling discrete: In this approach, the transmission duration τ_i' is calculated as follows.

$$\tau_i' = \begin{cases} t_B & \text{if } \tau_i \geq t_B, \\ t_1 & \text{if } \tau_i \leq t_1, \\ \min\limits_{j \in 1,\ldots,B} \{t_j : t_j \geq \tau_i\} & \text{otherwise.} \end{cases}$$

Floor discrete: The transmission duration τ'' is calculated as

$$\tau_i'' = \begin{cases} t_B & \text{if } \tau_i \geq t_B, \\ t_1 & \text{if } \tau_i \leq t_1, \\ \max\limits_{j \in 1,\ldots,B} \{t_j : t_j \leq \tau_i\} & \text{otherwise.} \end{cases}$$

In general, it is non trivial to theoretically analyze how this rounding procedure affects the energy efficiency of the optimal transmission schedule as well as the maximum and average delay suffered by packets. In the next section, however, we show through simulation that both these approaches suffer in either one of two metrics: energy consumed per packet or average/maximum delay suffered by any packet as compared to the optimal offline algorithm.

Definition 1 *An optimal discretized algorithm is defined as one that achieves the minimal energy when packets that arrive in the time interval $[0, T)$ are scheduled to complete transmission at one of the available discrete bandwidth levels in time at most T.*

Our discretized version of the offline algorithm combines the two approaches described above. It proceeds exactly as the optimal offline algorithm except when assigning individual transmission durations. The values of m_i and k_i are as calculated as in the optimal offline algorithm in Equations 1 and 2. Let B be the total number of discrete bandwidths available, p be the packet size, and $b_i, i = 1, \ldots, B$ be the individual bandwidths available. In addition to k_i, we define m_i', m_i'' and k_i' as follows.

$$\begin{aligned} m_i' &= \min_{j=1,\ldots,B} \left\{ \frac{p}{b_j} : \frac{p}{b_j} \geq m_i \right\} \\ m_i'' &= \max_{j=1,\ldots,B} \left\{ \frac{p}{b_j} : \frac{p}{b_j} \leq m_i \right\} \\ k_i' &= \begin{cases} k_i \cdot \frac{m_i - m_i''}{m_i' - m_i''} & \text{if } m_i' \neq m_i'', \\ m_i' & \text{otherwise.} \end{cases} \end{aligned}$$

Definition 2 *Let τ^* be the schedule defined as*

$$\tau_i^* = \begin{cases} m_j' & \text{if } k_{j-1} < i \leq k_{j-1} + k_{j-1}', \\ m_j'' & \text{if } k_{j-1} + k_{j-1}' < i \leq k_j. \end{cases}$$

It is easy to see that the schedule $\vec{\tau}^*$ produces an optimal offline schedule for a node with a fixed set of discretized bandwidth levels.

3.3 Simulation Results

We now compare the performance of our discretized offline algorithm to both the continuous version and the two naive rounding approaches through simulation. In these simulations, we model the packet arrivals as a Poisson process with a maximum arrival rate of 100 packets/sec. We arbitrarily choose a set of ten evenly spaced bandwidth levels such that an individual packet can be transmitted for a duration between 10 and 100 ms in increments of 10 ms. (This corresponds to bandwidths of approximately 1 Mbps to 100 Kbps for a 10-Kbit packet.) The energy required to transmit a packet with duration τ_i is given by

$$\omega(\tau_i) = 10^4 \frac{\tau_i}{0.06} \left(2^{\frac{0.12}{\tau_i}} - 1 \right). \tag{3}$$

We deliberately select the same power function as Prabhakar *et al.* [18, eqn. 17] to enable a straightforward comparison, but return in Section 6 to discuss its derivation and applicability to various application domains.

For simulations comparing multiple algorithms, each algorithm is run against the same set of packet arrivals. In other words, we first generate random packet arrival traces from the specified distribution for each offered load. We then feed these traces into each algorithm. Hence, while the actual traffic arrival pattern is random, it is fixed for any particular load value to allow for straightforward comparisons across algorithms. All graphs of simulation results represent the average of 20 independant trials.

We consider average power as the metric of merit when comparing these algorithms. Although energy per packet captures the difference in energy consumed between various algorithms that adaptively modify the transmission rates, it does not intuitively convey the power consumption during a scheduling interval, especially when considering radios that may not be on all the time. When transmitting packets at a faster rate, the total transmission obviously completes in less time. If $t \leq T$ is the total time required to schedule all the M packets that arrived in an interval $(0, T]$, then average power is simply $\omega(\vec{\tau})/t$ where $\omega(\vec{\tau})$ is total required energy to transmit all M packets.

The results of our simulations are shown in Figures 2 and 3. In Figure 2, we see that by flooring the transmission duration obtained in the optimal continuous algorithm, the transmission schedule consumes far more power than that computed by the optimal continuous algorithm. Conversely, rounding the transmission durations up with the Ceiling discrete algorithm takes less power, but can introduce significant additional delay, as seen in Figure 3. The points when the ceiling discretized optimal offline algorithm is close to the rest of the graphs corresponds to the case where the rounding results in no extra penalty. For example, if the continuous version assigned a transmission duration of exactly 20 ms, then there is no rounding loss. While these effects are to be expected and easy to calculate analytically, we plot them here to provide a visual comparison for the optimal discretized algorithm. Our optimal discretized offline represents a trade-off between power consumption and delay and, hence, represents a closer approximation to the continuous case. We use the optimal discretized offline algorithm as a lower bound for comparisons with the online algorithms presented in the next section.

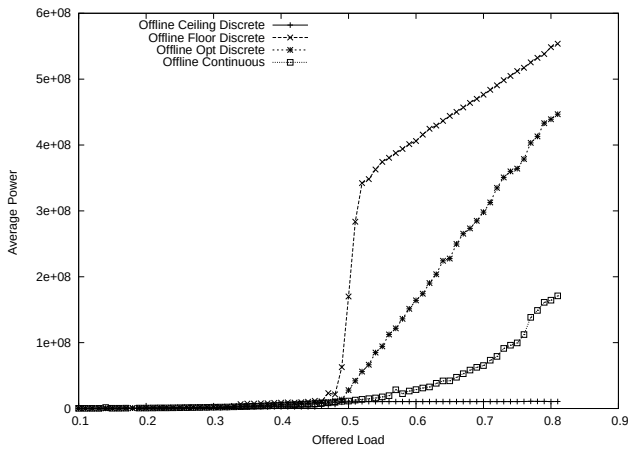

Figure 2: Average power comparison between Ceiling discrete, Floor discrete, and the optimal discrete offline algorithms with Poisson arrivals.

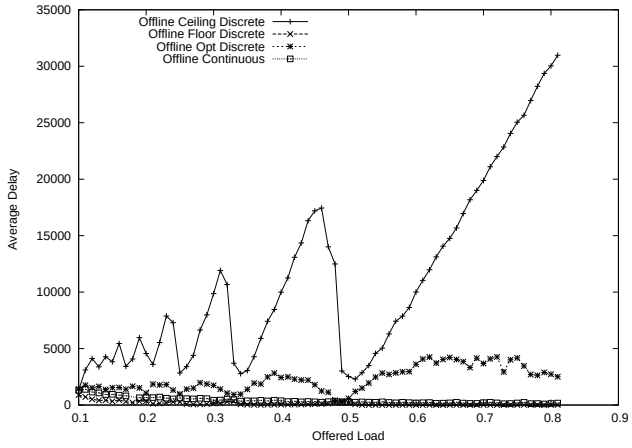

Figure 3: Average delay per packet for Ceiling discrete, Floor discrete and the optimal discrete offline algorithms with Poisson arrivals.

4. ONLINE ALGORITHMS

So far, we have considered offline algorithms with knowledge of the upcoming arrivals. In general, however, a node does not know upcoming arrivals in advance. In this section, we consider an extremely simple online algorithm which we term a *delay look-ahead* algorithm. We show through simulation that the delay look-ahead algorithm achieves performance close to the discretized version of the offline algorithm despite its simplicity.

4.1 Delay look-ahead algorithm

In order to reasonably approximate the offline algorithm, we need some way to predict upcoming arrivals—a difficult problem in general. One way to obviate the need for this information is to maintain a buffer of recent arrivals and schedule them at a later time. Clearly, there is a trade-off involved when sizing the buffer: buffering too many packets could lead to intolerable delays, while buffering too few reduces the potential for power savings. Regardless of the buffer size, the scheduling problem reduces to offline scheduling of the set of packets received in the previous interval.

The algorithm operates in two steps: a look-ahead stage, and a scheduling stage. The algorithm uses a configurable parameter

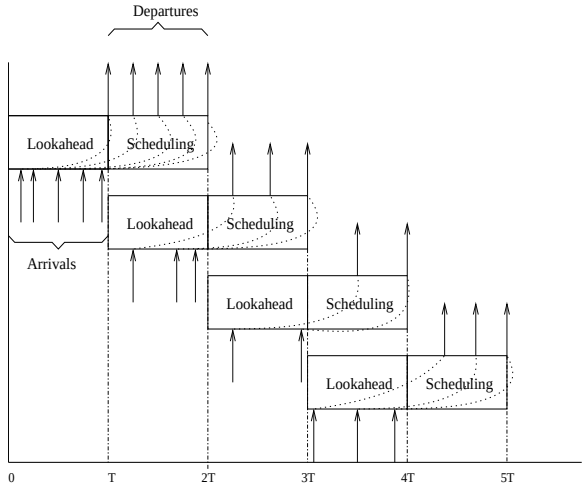

Figure 4: The two phases of the online delay look-ahead algorithm: packets buffered in a look-ahead phase are transmitted in the following scheduling phase.

called the *look-ahead*, D (also called a time horizon in the literature [28]). The algorithm buffers all arrivals in the interval $[t, t+D)$ in the look-ahead stage and transmits these packets in the next interval $[t+D, t+2D)$ in the scheduling stage, while collecting the arrivals in this interval to be scheduled in the next. This process is illustrated by Figure 4.

From the optimal offline algorithm, it is easy to see that the most efficient way to schedule a set of packets that all arrive at or before time 0 is to divide all the packet transmissions equally over the interval $[0, T)$. Any other way of transmitting, say, sending one packet very slow and the others faster would by the definition of convexity take more energy than when all the packets are sent at the same transmission duration.

The problem can be formally stated as follows. If $a_1, a_2, \ldots, a_n$ are the n arrivals in the time interval $[t, t+D)$, then the optimal transmission duration for of these packets given continuously variable bandwidths is given by

$$\begin{aligned} m &= \frac{D}{n} \\ \tau_i &= m \quad \forall\, a_i, i \in 1, \ldots, n. \end{aligned}$$

4.2 Simulation results

Figures 5 and 6 compare the continuous bandwidth version of the delay look-ahead algorithm with look-ahead values of 500, 1000, and 2000 ms for Poisson arrivals. We observe that energy efficiency increases with the look-ahead parameter and decreases with increasing load, as expected. In practice, however, traffic arrival patterns may be far more bursty. In Figures 7 and 8, we repeat the simulation using the 'bursty' arrival process of Uysal-Biyikoglu *et al.* [28]. The inter-arrival times between the packets are i.i.d with

$$\Pr[D_i = a_1] = \beta = 1 - \Pr[D_i = a_2],$$

where a_1, a_2 and β are parameters. When a_1 is small and β is large, the arrivals tend to be bursty with a high probability. In our experiments, we fix β at 0.9, $a_2/a_1 = 9$ and $\lambda_{max} = 1$. We vary λ between 0.1 and 0.8 and plot the average power and per-packet delay as a function of λ.

From Figure 7, we can see that look-ahead settings of both 2000 and 1000 ms perform similarly to the offline optimal in terms of energy efficiency. The 500 ms setting becomes inefficient as the load

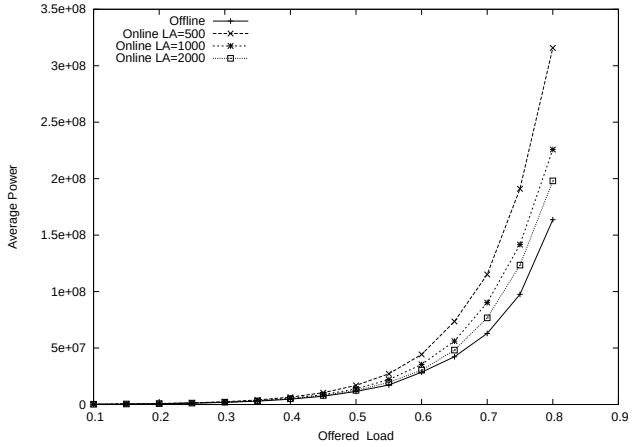

Figure 5: Average power for both online and offline algorithms under Poisson arrivals.

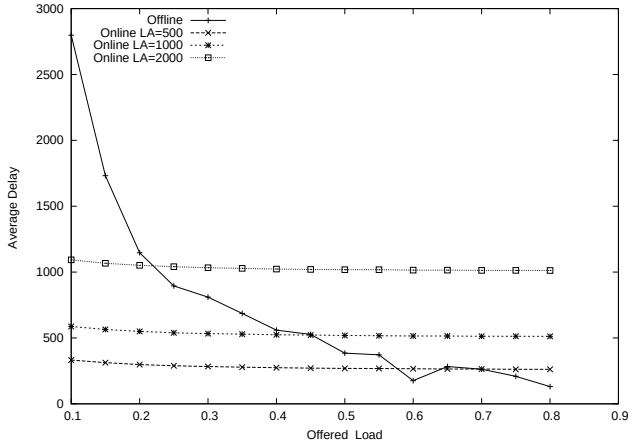

Figure 6: Average packet delay for both online and offline algorithms under Poisson arrivals.

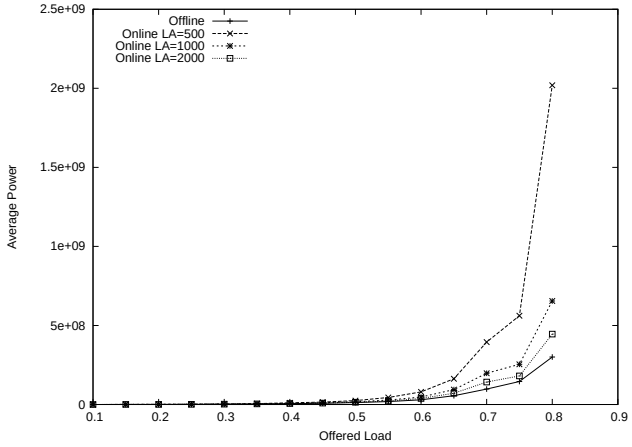

Figure 7: Average power comparison between online and offline algorithms under bursty arrivals.

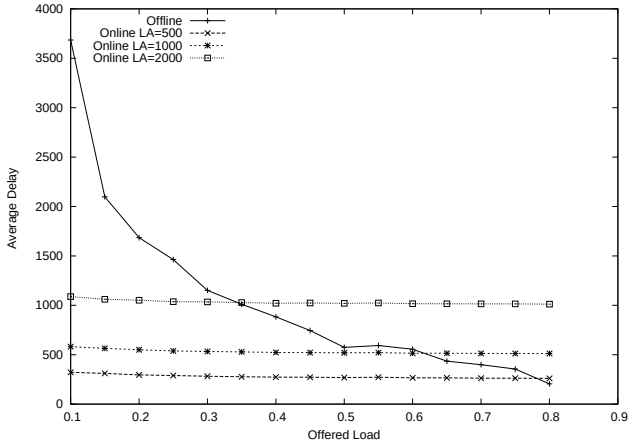

Figure 8: Average packet delay for both online and offline algorithms under bursty arrivals.

increases, however. At high loads, the bursts become larger than the look-ahead buffer, forcing the algorithm to schedule all the arrivals at the highest possible bandwidth level; the look-ahead phase is not long enough to take advantage of the lull period following the burst.

One observation from the delay graphs (for both Poisson and bursty arrivals) is that the average packet delay doesn't depend on the offered load. This independence is due to the fact that all packets (assuming the offered load is less than one) arriving in the look-ahead phase get scheduled in the immediately following scheduling phase. Although our algorithm does not allow the specification of individual packet deadlines, the stability in average delay may make it attractive for applications that cannot tolerate arbitrary packet delays or large variance in inter-packet arrival times (jitter).

4.3 Discretized delay look-ahead algorithm

Unfortunately, the optimal transmission rate for a particular interval may not be possible for any given radio. Hence, we have to apply a discretization function similar to those shown in Section 3 to conform to the constraints imposed by commodity radios. Intuitively, we divide the packets among the two bandwidths closest to the computed optimum. Formally, if $b_i, i \in 1, \ldots, B$ are the bandwidth levels that the radio supports, then the possible transmission durations for a packet size p can be computed as before as $t_i = p/b_i$. The transmission durations τ_i' for packets in the discretized setting can then be calculated as follows. Let

$$m' = \min_{j=1,\ldots,B} \left\{ \frac{p}{b_j} : \frac{p}{b_j} \geq \tau \right\} \quad (4)$$

$$m'' = \max_{j=1,\ldots,B} \left\{ \frac{p}{b_j} : \frac{p}{b_j} \leq \tau \right\} \quad (5)$$

$$n' = \frac{D - n \cdot m''}{m' - m''} \quad (6)$$

and

$$\tau_i' = \begin{cases} m' & \text{if } 1 \leq i < n', \\ m'' & \text{if } n' \leq i < n. \end{cases} \quad (7)$$

Equation 4(5) finds the transmission duration $m'(m'')$ that corresponds to an available bandwidth level that is just above(below) the actual duration τ assigned by dividing the interval continuously between all the arrivals. Equation 6 finds an optimal number of departures that should be assigned a transmission duration of m' so that all the packets can be scheduled within the interval D. The schedule τ thus splits the next interval $[t+D, t+2D)$ as equally as possible among all the arrivals in the previous interval. Despite its

simplicity, simulations show that this algorithm is efficient both in terms of power consumption and per-packet delay.Above all, it is extremely We defer presenting simulation results for this algorithm until Section 6, however, where we discuss the impact of bandwidth level selection.

5. DISTRIBUTED DELAY LOOK-AHEAD

The algorithms presented so far deal only with the case of a single node with exclusive access to the communications channel. In steady state, a node will adapt its transmission schedule so that packets are transmitted at roughly the same rate as their arrival, but at the lowest bitrate that accomodates the offered load. The channel is therefore occupied a great deal of the time. In a multi-node broadcast network, such oblivious node behavior may lead to increased queue sizes at other nodes causing them to transmit at a much higher bandwidth level in the next interval. This oscillatory behavior would likely negate any potential energy savings due to lazy scheduling. For the greedy power-savings scheme presented in the previous sections to work efficiently, we require a mechanism for nodes to estimate the total load on the channel so that each node can independently compute its share of the channel capacity and adjust its transmission rate accordingly. In this section, we propose an extension to the CSMA/CA MAC algorithm that is based on our online delay look-ahead algorithm.

Before proceeding with the algorithm, we state several architectural assumptions, each of which we believe is easily met in today's sensor networks.

- All nodes are not malicious and use a CSMA/CA-based channel access protocol with our extensions. All nodes use a single look-ahead parameter (interval duration) that is specified *a priori*.
- The network is an arbitrary collection of nodes communicating in a many-to-many fashion using a shared, broadcast medium. We assume the required power level for each bitrate is the same for all destinations. (Known techniques to gain efficiency by considering different power functions per destination are computationally quite expensive [7].)
- Nodes can be time synchronized up to millisecond precision (perhaps through referenced-broadcast-based techniques proposed by Elson *et al.* [8]).
- Each node can determine both the sender and the duration of any packet transmitted over the channel. While many MAC protocols currently export this information, some may require the addition of a small header.

Our distributed algorithm proceeds in exactly the same fashion as the discretized online algorithm for the single node case: each node operates in two phases, the look-ahead phase and the scheduling phase.The distributed algorithm differs from the single-node algorithm only in the scheduling phase; the look-ahead phase proceeds exactly as in the case of the single node. In the look-ahead phase, all the packets that arrive in the current interval are queued up along with any unsent packets from the previous scheduling interval. In the the single node case, the scheduling interval is divided equally among all these packets. In the distributed case, the scheduling interval is divided among the locally queued packets and an estimate of the packets queued at the other nodes sharing the channel.

Pseudocode for our distributed algorithm is shown in Figure 9. The scheduling phase proceeds as follows. (For simplicity we assume all the packets are of equal size. This can be easily extended to packets of unequal size.)

```
Initialize :
currentTime = 0;
count = packet arrivals till lookahead time T;
availableTime = T;
nextIterationStartTime = T;

repeat {
while (currentTime <= nextIterationStartTime) {
  if (channel is Idle) {
    current Packet Duration=discretize(availableTime/count);
    send Packet;
    wait for ack;
    if (no_ack) {  // collision
      wait randomTime in (0, 2 * Last CW Value);
      availableTime -= randomTime;
    }  else { // no collision
      count --;
      availableTime -= currnetPacketDuration;
    }
  } else {
    wait till Channel is free;
    if (transmitter of the last packet is transmitting
        for the first time in this iteration) {
      if ( myself transmitted atleast one packet in
           this iteration ) {
        // my load is already factored in
        count = availableTime / tranmsissionDuration;
      } else  {
        // my load is not factored in
        count = availableTime / transmissionDuration
                + myNumberofPackets;
      }
    } else {
      count --;
    }
    availableTime -= last packet transmission duration;
  }
}
nextIterationStartTime += T;
update count with new arrivals;
}
```

Figure 9: Pseudocode for the distributed online algorithm.

1. Initially, the node that gets access to the channel aggressively reduces its transmission rate assuming that it has access to the entire channel capacity.

2. Nodes that have packets to send snoop the channel to determine the rate of the packet currently being transmitted. Using this rate, listening nodes estimate the load at the transmitting node. Note that these nodes would be listening to the channel anyway, awaiting the end of the current transmission.

3. If the transmitting node is using the channel for the first time in this interval, each node (other than the transmitting node) increases its estimate of number of packets that need to be sent in the current interval by its estimate of the transmitting node's load.

After each node with packets to send transmits at least one packet, every node's estimate of the load on the channel converges to the correct value and for the remainder of the time interval. Each node selects its transmission speed such that the channel is split equally among all the queued packets. The nodes also take any CSMA/CA channel-access delays (resulting from collisions, forced back-offs, etc.) into account when determining the transmission rate of its own packets.

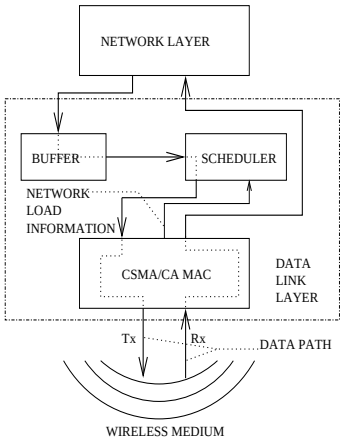

Figure 10: L-CSMA/CA MAC architecture.

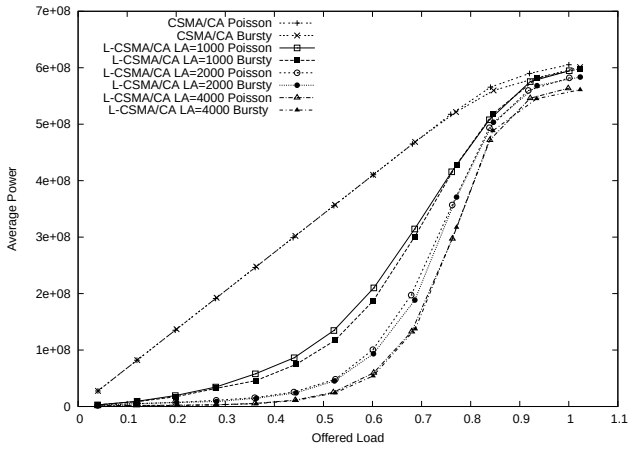

Figure 11: The average total power consumption of a four-node network using L-CSMA/CA and CSMA/CA with varying Poisson and bursty loads. We show three different values of the look-ahead parameter, 1000, 2000, and 4000 ms.

5.1 Online implementation using CSMA/CA

Our online implementation of the distributed delay look-ahead algorithm consists of a new layer between the network layer of the protocol stack and the MAC layer, which we call the Scheduling/Buffering layer. The extended protocol stack is shown in Figure 10. Our layer consists of a buffer for storing the packets that arrive in the look-ahead phase of the algorithm and a scheduler that assigns transmission times and durations for each of the queued packets. The scheduler uses feedback from the underlying MAC layer to estimate the current load on the channel to assign schedules for packets collected in the look-ahead phase. We call this new, extended MAC Look-ahead CSMA/CA, or L-CSMA/CA for short.

CSMA/CA (carrier sense, multiple access, with collision avoidance) is widely accepted as a standard for medium access in wireless networks, including in the IEEE 802.11 standard [10]. The CSMA/CA protocol works similarly to the CSMA/CD standard used in Ethernet (802.3). When a node wants to transmit data, it first checks to see if the channel is busy. If it is, it continuously senses the channel, waiting for it to become idle. When the channel becomes idle, the node first waits for a specified inter-frame spacing, then sets a contention timer for a delay uniformly selected in the interval $[0, CW]$, where CW is a predefined contention-window length. When this timer expires, the node transmits a packet and waits for the receiver to send an ACK. If no ACK is received, the packet is assumed lost to collision, and the source node tries again, choosing a contention back-off selected uniformly at random from an interval twice as long as the one before. If, during the back-off period, the node senses that another station has begun transmission, it does not reset its timer but freezes it instead and restarts it when the packet completes transmission. In this way, a station that happens to choose a longer timer value gets higher priority in the next round of contention. A full analysis of CSMA and variants can be found in Kleinrock and Tobagi [14].

Our scheduler needs to extract several pieces of information from the MAC layer. In particular, the scheduler gets feedback from the CSMA/CA layer about each of the following that are factored into the scheduling algorithm by snooping the broadcast channel.

- The source, duration, and rate of any detected packet transmission on the wireless channel
- The length of any back-off intervals induced due to the collision avoidance algorithm

It is worth noting that this algorithm does not necessarily affect the channel capacity. At first blush, it might appear that decreased packet transmission speeds might lead to an increase in channel contention and collisions, thereby reducing the channel throughput compared to when each node transmits packets at the maximum possible rate. While L-CSMA/CA can lead to increased channel contention as compared to CSMA/CA, the resulting lost time due to collisions is fed back into the scheduler at the sending nodes. The schedulers use this feedback to determine the transmission duration of the next attempted packet transmission. The algorithm automatically compensates for increased collision rates by speeding up the next set of transmissions. Due to this constant adaptation to collision losses, the throughput remains competitive with that of regular CSMA/CA, although the energy efficiency of the scheme may decrease as the channel contention increases; in the worst case, all nodes send at the maximum rate, which is precisely CSMA/CA.

5.2 Simulation results

We have implemented both CSMA/CA and L-CSMA/CA in our simulator. CSMA/CA has several configurable parameters including $\alpha = PROP/TRANSP$ and CW. Here, $PROP$ refers to the maximum propagation delay between any two nodes and $TRANSP$ is the transmission delay for an average-sized packet. For wireless channels, α is around 0.1 [13]. We chose CW to be twice the size of $PROP$ and the maximum binary exponential back-off exponent to be five. So, the maximum time a node waits before transmitting a packet once it senses the channel is idle is $31 \cdot CW$. $PROP$ has been chosen as 0.5 ms.

To evaluate how L-CSMA/CA compares to CSMA/CA in terms of energy efficiency, we simulate a network of four nodes, each with equal load. The total power consumption of the network is plotted as a function of total offered load (the sum of the load of each of the four nodes) for three different look-ahead parameter values in Figure 11. The X axis shows the aggregate load over four nodes and the Y axis shows the total average power. We compare CSMA/CA transmitting at maximum speed with the L-CSMA/CA algorithm using a continuous bandwidth distribution. The greatest savings occur at low levels of network utilization. For example, at a load of 0.2 L-CSMA/CA with $D = 4000$ ms consumes 99% less energy than CSMA/CA. The L-CSMA/CA results are a lower bound on the potential energy savings, however; the achiev-

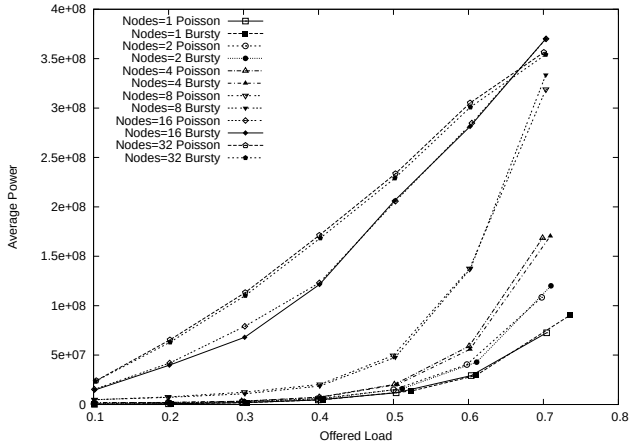

Figure 12: Energy efficiency of L-CSMA/CA for 1-, 2-, 4-, 8-, 16-, and 32-node networks under Poisson and bursty loads.

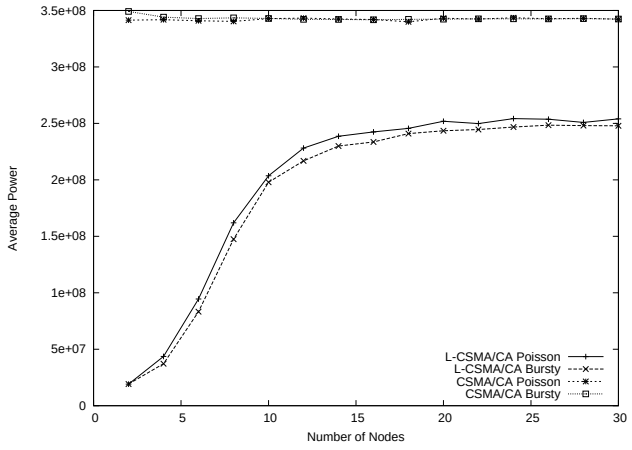

Figure 13: Total average power consumed by L-CSMA/CA and CSMA/CA networks with a varying number of nodes but constant aggregate Poisson and bursty loads.

able savings with a particular set of bandwidths depends greatly on the distribution of available bitrates as discussed in Section 6.

From Figure 11, we can seen that the power consumption of CSMA/CA is almost linear with respect to load, which is expected since, as the load increases, the number of packets per unit time increases proportionally. Since each packet consumes the same amount of energy, averge power increases proportionally. On the other hand, our L-CSMA/CA algorithm yields a convex function related to the convexity of the power function. As load increases, our algorithm adaptively approaches the maximum possible bandwidth level.

L-CSMA/CA degrades gracefully as the number of nodes in the network increases. The overhead involved in discovering the actual load network increases with n: The distributed schedule sends packets at inappropriately low speeds until each node has had a chance to access the channel. Hence, care must be taken to select a look-ahead interval appropriate for the size and load of the network. As can be seen in Figure 12, for a fixed look-ahead parameter (4000 ms) and number of nodes, the effectiveness of L-CSMA/CA gracefully degrades to CSMA/A.

Figure 13 fixes the aggregate load to 0.6 and instead varies the number of nodes in the network between 2 and 32. Each node has

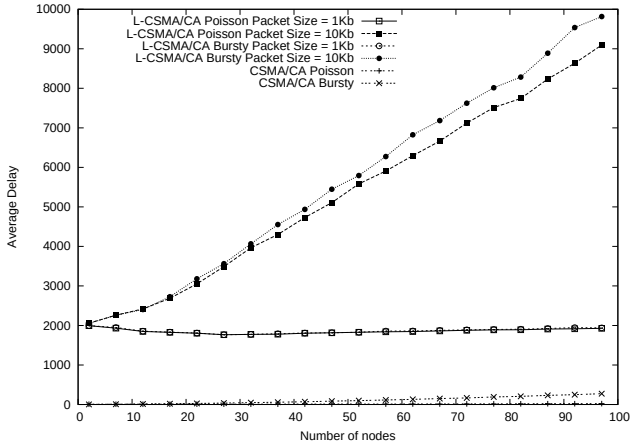

Figure 14: Average per-packet delay comparison between L-CSMA/CA and CSMA/CA with a varying number of nodes but constant aggregate Poisson and bursty loads.

an independent set of packet arrivals drawn from a Poisson distribution with $\lambda = 0.6/n$, where n is the number of nodes. The same set of packet arrivals was used for both the L-CSMA/CA and CSMA/CA algorithms for each value of n. From the simulations, it appears that the energy efficiency of L-CSMA/CA degrades rapidly with the number of nodes, again due to the initial overhead incurred while each node conveys their load information to the others. After about 10 nodes, however, the difference in power consumption between L-CSMA/CA and CSMA/CA becomes relatively stable at about 20–25%.

Finally, Figure 14 plots the average per-packet delay for both L-CSMA/CA and CSMA/CA networks with increasing numbers of nodes. In this figure, we have used an L-CSMA/CA with 2000 ms look-ahead value, but two different packet sizes: 10Kb, as before, and 1Kb. In the 10-Kb case, L-CSMA/CA suffers from much higher per-packet delay compared to CSMA/CA because of both the 2000-ms look-ahead buffer and the increase in initial contention as the number of nodes increases—as the number of nodes increases past 12, the nodes are no longer able to fully deplete their buffers by the end of the interval, leading to an unbounded per-packet delay. If the size of the packet size is decreased to 1 Kb, however, the load discovery period consumes a much smaller portion of the look-ahead interval, and each nodes is able to transmit all of its queued packets in each interval. Determining the appropriate look-ahead buffer size for a given MTU and network size is a subject of future work. We are also exploring mechanisms for networks to increase the minimum allowable bandwidth setting when nodes failed to fully deplete their queues in the previous interval.

We presented a theoretical argument at the end of Section 5.1 for the throughput efficiency of L-CSMA/CA. Here, we validate that argument through simulation, showing that, for small networks, the throughput of L-CSMA/CA remains extremely close to that of CSMA/CA even when the total aggregate traffic approaches the channel capacity. Table 1 presents the throughput of a four-node network using both CSMA/CA and L-CSMA/CA for varying levels of offered load. The values were calculated by simulating the algorithms on the same set of Poisson arrivals. We measure throughput as the total number of packets transmitted divided by the total time required by all the packets to be transmitted times the maximum radio bandwidth. Hence, the actual offered load differs slightly from the λ parameter of the Poisson distribution.

Offered Load	L-CSMA/CA	CSMA/CA
0.039796	0.038447	0.039789
0.121036	0.115578	0.121024
0.238726	0.234020	0.238680
0.319413	0.307116	0.319379
0.402544	0.394611	0.402472
0.487580	0.472817	0.487422
0.563117	0.546648	0.563004
0.642413	0.629806	0.642284
0.723143	0.708923	0.723064
0.761976	0.739715	0.761724
0.841525	0.825013	0.831241
0.926280	0.873536	0.865040
1.003830	0.896219	0.893350

Table 1: Observed throughput for a network of four nodes using L-CSMA/CA and CSMA/CA for different offered loads with a Poisson arrival distribution.

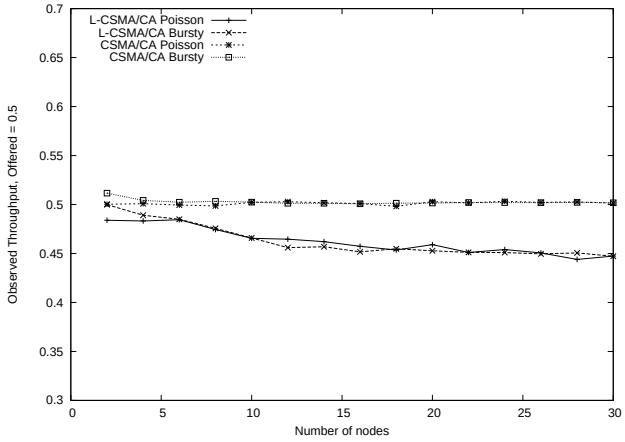

Figure 15: Throughput comparison between L-CSMA and CSMA for an offered total load of 0.5.

As can be seen in the table, the efficiency of both CSMA/CA and L-CSMA/CA reduce as the offered load increases. When the load increases, the number of collisions increase leading to lower throughput. The throughput of L-CSMA/CA is very close to that of the CSMA/CA across all load levels, however. L-CSMA/CA has slightly lower throughput as compared to the CSMA/CA due to its load discovery phase. Unfortunately, the effect grows larger with increasing network size. Figure 15 plots the observed throughput of L-CSMA/CA in comparison with CSMA when we vary the number of nodes keeping the overall load constant at 0.5. It can be observed that as the number of nodes increase, the efficiency of L-CSMA/CA drops by about 10% with 32 nodes. This inefficiency is due to a larger contention for the channel (in the initial phase of the algorithm where the load estimates have not stabilized) as opposed to the regular CSMA/CA where the packets are transmitted as fast as possible. This effect can be ameliorated in the same fashion as the delay in Figure 13 by selecting an appropriate look-ahead buffer for the packet length and network size.

6. DISCUSSION

There are several issues that arise when deloying L-CSMA/CA—or any lazy scheduling technique—in physical a radio; we consider two of the most significant here. First, we examine the impact of the distibution of available transmission rates for a particular tranceiver. Second, we consider the applicability of our chosen simulation parameters by deriving the transmission power function, $\omega(\tau)$, defined in Equation 3.

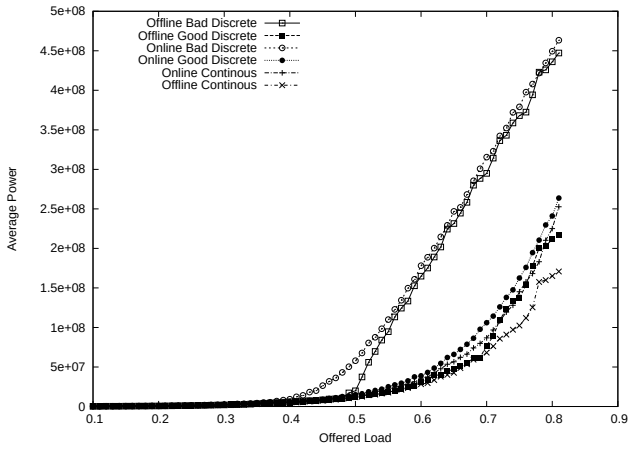

Figure 16: Energy per unit time for both online and offline algorithms with Poisson arrivals for varying transmission speed sets.

6.1 Transmission rate selection

Figure 2 hints at the effect that round-off errors from bandwidth discretization can have on the power consumption of a given transmission schedule. We postponed further discussion of discretization effects in later sections by presenting results for the continuous bandwidth case. We now demonstrate the dramatic effect a poor selection of bandwidth rates can have on transmission efficiency.

In Figure 16, we reproduce the results of the optimal discretized offline algorithm ('offline discrete') from Figure 2, the offline optimal algorithm ('offline continuous'), and online continuous algorithm ('online continuous') with a delay look-ahead buffer of 2000 ms from Figure 5. This time, however, we discretize both the online and offline algorithms using two different choices of available bit rates. The first set consists of bandwidths that correspond to a transmission durations of 10, 20, ..., 100 ms in increments of 10 ms for a 10-Kbit packet. The second set consists of bandwidths that would have a transmission duration of either 10, 11, 14, 17, 20, 25, 40, 60, 80, or 100 ms. Notice that we deliberately keep the choice of lowest and highest bandwidths the same. For reasons that are obvious from the figure, we call the first set 'bad' and the second set 'good.'

As can be seen in Figure 16, the 'good' set performs much better than the 'bad' set as far as energy savings are concerned, especially at high loads. Hence, care should be taken when designing a multi-rate radio to support efficient lazy packet scheduling. The obvious conclusion given the convex power function is to provide a larger number of high-bandwidth options than low ones. Unfortunately, the speeds offered by current radio technologies (e.g., 802.11a [10] and the CC1000 RF transceiver used in the Berkeley Motes [4]) are often almost the opposite. A detailed discussion of the issues involved in designing the appropriate set of encodings are outside the scope of this paper.

Similarly, different radios expose differing numbers of transmission power levels. Hence, despite the availability of multiple speeds, it may be the case that the same transmission power is required for a set of speeds. In that case, the maximum speed for the minimum required power level should be used. It is straightforward

to extend our quantization method to account for this factor. Certain existing MAC technologies (802.11 being the prime example), however, require a packet preamble to be transmitted at maximum possible power to avoid hidden terminal problems. We have not yet investigated the impact this requirement would have on our power savings scheme.

Unfortunately, the transmission power curve is often discontinuous at zero. Namely, many radios consume significant power when idle. In this case, it may be beneficial to transmit all packets at the highest rate in order to turn off the radio at the earliest possible moment. This is not always a viable scenario, however, since the node many need to remain awake to listen for transmissions from other nodes. Hence, lazy packet scheduling may continue to be beneficial even for these devices.

6.2 Convexity in different speed ranges

We now derive the power function used in our simulations and show its applicability over a range of tranmission speeds. It is well known (see, for example, Cover and Thomas [5]) that for an AWGN channel with average signal power constraint P and noise power constraint N, an optimal channel coding scheme that employs randomly generated codes approaches the Shannon channel capacity given by

$$C = \frac{1}{2}\log_2\left(1 + \frac{P}{N}\right) \text{bits/transmission}.$$

We assume a slightly sub-optimal code with rate $R = \alpha C$, where $\alpha \approx 1$. Substituting for C in the above equation and solving for P, we get

$$P = N\left(2^{2\alpha R} - 1\right).$$

Now, let us consider 1 Mbps channel with $R = 6$ [18]. One realization of such a channel is a radio with a symbol (baud) rate of 1/6 MHz that transmits at most 6 bits per symbol [24]. Assuming the transmission duration of a 10Kb packet is τ, R and τ are inversely related as follows,

$$R = \frac{0.01 \cdot 6}{\tau}.$$

Therefore, energy per bit $\omega = P \cdot 1/R$, and energy per packet ω_{1Mbps} assuming a noise of 1, is given by

$$\omega_{1Mbps} = 10^4 \frac{\tau}{0.06}\left(2^{\frac{0.12}{\tau}} - 1\right).$$

This is exactly Equation 3, where τ ranges from 0.01 s (1 Mbps) to 0.1 s (100 Kbps).

Many modern sensor networks, however, operate at much lower speeds; 10 Kbps–100 Kbps is typical. The production version of the CC1000 RF transceiver used in the MICA2 motes, for example, is capable of rates from 0.3 to 76.8 Kbps [4]. Assuming a maximum channel bandwidth of 100 Kbps, the theoretical power to transmission duration relation would be

$$\omega_{100Kbps} = 10^4 \frac{\tau}{0.6}\left(2^{\frac{1.2}{\tau}} - 1\right).$$

τ varies from 0.1 s (100 Kbps) to 1 s (10Kbps) in this equation. Adjusting for the new values of τ, $\omega_{100Kbps}$ is exactly the same as ω_{1Mbps}. Hence, our simulation results are equally valid in the 10–100 Kbps range: the shape of the power curve remains the same.

7. CONCLUSIONS

There has been a great deal of research into lazy packet scheduling techniques. To the best of our knowledge, however, we are the first consider applying them to the shared, broadcast channels commonly found in sensor network environments. The optimal per-packet transmission rate in a broadcast network is governed in practice by the available bit rates of the given radio(s), any per-packet deadlines, and the offered load at every node contending for the shared channel. We have shown how to efficiently discretize the known, optimal offline algorithm schedule for an individual node using a private channel given a set of available, discrete transmission rates. We then derived an online approximation of this algorithm and showed that it is more efficient than any deterministic transmission schedule. Finally, we presented an extension to traditional CSMA/CA MAC scheme, L-CSMA/CA, that allows multiple nodes to estimate the current demand for the wireless channel and adjust their transmission schedules accordingly. L-CSMA/CA provides improved energy efficiency for a set of uncoordinated wireless nodes without significantly impacting the capacity of the shared broadcast channel.

An interesting question for future work is to consider minimizing not just the total transmission energy in the network, but the total *communication* energy, including the energy spent receiving packets. Over time, receive power is expected to become an increasingly greater fraction of the total wireless power budget. We do not yet understand how lazy packet scheduling schemes like L-CSMA/CA affect non-transmitting nodes that might be forced to spend additional time listening to the channel.

Acknowledgements

This paper benefitted greatly from discussions with Geoff Voelker, Stefan Savage, and Rajesh Gupta, and comments from the anonymous reviewers and our shepherd, Mani Srivastava.

8. REFERENCES

[1] ADLER, F. P. Minimum energy cost of an observation. *IEEE Transactions on Information Theory 1*, 3 (Dec. 1955), 28–32.

[2] CHEN, B., JAMIESON, K., BALAKRISHNAN, H., AND MORRIS, R. Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. *ACM Wireless Networks Journal 8*, 5 (Sept. 2002), 481–494.

[3] CHEN, J.-C., SIVALINGAM, K. M., AGRAWAL, P., AND KISHORE, S. A comparison of MAC protocols for wireless local networks based on battery power consumption. In *Proc. IEEE Infocom* (San Francisco, California, Mar. 1998), pp. 150–157.

[4] CHIPCON AS. SmartRF CC1000 product notice high data rate (rev 1.0), Dec. 2001.

[5] COVER, T., AND THOMAS, J. *Elements of Information Theory*. John Wiley & Sons, New York, 1991.

[6] CROSSBOW TECHNOLOGY, INC. MICA2 Mote (MPR400CB), 2003.

[7] EL GAMAL, A., NAIR, C., PRABHAKAR, B., UYSAL-BIYIKOGLU, E., AND ZAHEDI, S. Energy-efficient scheduling of packet transmissions over wireless networks. In *Proc. IEEE Infocom* (New York, New York, June 2002), pp. 1773–1782.

[8] ELSON, J., GIROD, L., AND ESTRIN, D. Fine-grained network time synchronization using reference broadcasts. In *Proc. 5th USENIX Symposium on Operating Systems Design and Implementation* (Boston, Massachusetts, Dec. 2002), pp. 147–163.

[9] FEENEY, L. M., AND NILSSON, M. Investigating the energy consumption of a wireless network interface in an ad hoc

networking environment. In *Proc. IEEE Infocom* (Anchorage, Alaska, Apr. 2001), pp. 1548–1557.

[10] INSTITUTE OF ELECTRONIC AND ELECTRICAL ENGINEERS (IEEE). Wireless medium access control (MAC) and physical layer (PHY) specifications. Standard 802.11, 1999.

[11] IRANI, S., SHUKLA, S., AND GUPTA, R. Algorithms for power savings. In *Proc. ACM-SIAM Symposium on Discrete Algorithms* (Baltimore, Maryland, Jan. 2003), pp. 37–46.

[12] KAWADIA, V., AND KUMAR, P. R. Power control and clustering in ad hoc networks. In *Proc. IEEE Infocom* (San Francisco, California, Apr. 2003).

[13] KESHAV, S. *An Engineering Approach to Computer Networking*. Addison-Wesley, Reading, Massachusetts, 1997.

[14] KLEINROCK, L., AND TOBAGI, F. A. Packet switching in radio channels: Part I—carrier sense multiple-access modes and their throughput-delay characteristics. *IEEE Transactions on Communications COM-23*, 12 (Dec. 1975), 272–288.

[15] KULIK, J., HEINZELMAN, W. R., AND BALAKRISHNAN, H. Negotiation-based protocols for disseminating information in wireless sensor networks. *Wireless Networks 8*, 2–3 (March–May 2002), 169–185.

[16] LORCH, J. R., AND SMITH, A. J. Improving dynamic voltage scaling algorithms with PACE. In *Proc. ACM SIGMETRICS/Performance* (Cambridge, Massachusetts, June 2001), pp. 50–61.

[17] NUGGEHALLI, P., SRINIVASAN, V., AND RAO, R. R. Delay constrained energy efficient transmission strategies for wireless devices. In *Proc. IEEE Infocom* (New York, New York, June 2002).

[18] PRABHAKAR, B., UYSAL-BIYIKOGLU, E., AND EL GAMAL, A. Energy-efficient transmission over a wireless link via lazy packet scheduling. In *Proc. IEEE Infocom* (Anchorage, Alaska, Apr. 2001).

[19] PRICE, C. E., SIVALINGAM, K. M., CHEN, J.-C., AND AGARWAL, P. Power-aware scheduling algoritms for wireless networks. In *Proc. International Conference on Intelligent Computing and VLSI* (Kalyani, India, Feb. 2001).

[20] QUAN, G., AND HU, X. Energy efficient fixed-priority scheduling for real-time systems on variable voltage processors. In *Proc. Design Automation Conference* (Las Vegas, Nevada, June 2001), pp. 828–833.

[21] RAGHUNATHAN, V., GANERIWAL, S., SCHURGERS, C., AND SRIVASTAVA, M. B. E^2WFQ: An energy efficient fair scheduling policy for wireless systems. In *Proc. International Symposium on Low Power Electronics and Design* (Monterey, California, Aug. 2002), pp. 30–35.

[22] RAMANATHAN, R., AND HAIN, R. Topology control of multihop wireless networks using transmit power adjustement. In *Proc. IEEE Infocom* (Tel Aviv, Israel, Mar. 2000), pp. 404–413.

[23] SAPUTRA, H., KANDEMIR, M. T., VIJAYKRISHNAN, N., IRWIN, M. J., HU, J. S., HSU, C.-H., AND KREMER, U. Energy-conscious compilation based on voltage scaling. In *Proc. ACM SIGPLAN Joint Conference on Languages, Compilers, and Tools for Embedded Systems & Software and Compilers for Embedded Systems* (Berlin, Germany, June 2002), pp. 2–11.

[24] SCHURGERS, C., ABERTHORNE, O., AND SRIVASTAVA, M. B. Modulation scaling for energy aware communication systems. In *International Symposium on Low Power Electronics and Design (ISLPED)* (Huntington Beach, California, Aug. 2001), pp. 96–99.

[25] SCHURGERS, C., RAGHUNATHAN, V., AND SRIVASTAVA, M. B. Modulation scaling for real-time energy aware packet scheduling. In *Proc. IEEE Global Communications Conference* (San Antonio, Texas, Nov. 2001), pp. 3653–3657.

[26] SCHURGERS, C., AND SRIVASTAVA, M. B. Energy efficient wireless scheduling: Adaptive loading in time. In *Proc. IEEE Wireless Communications and Networking Conference* (Orlando, Florida, Mar. 2002), pp. 706–711.

[27] UYSAL-BIYIKOGLU, E., AND EL GAMAL, A. Energy-efficient packet transmission over a multi-access channel. In *Proc. IEEE International Symposium on Information Theory* (Lausanne, Switzerland, July 2002), p. 153.

[28] UYSAL-BIYIKOGLU, E., PRABHAKAR, B., AND EL GAMAL, A. Energy-efficient packet transmission over a wireless link. *ACM/IEEE Transactions on Networking 10*, 4 (Aug. 2002), 487–499.

[29] WEISER, M., WELCH, B. B., DEMERS, A. J., AND SHENKER, S. Scheduling for reduced CPU energy. In *Proc. 1st USENIX Symposium on Operating Systems Design and Implementation* (Monterey, California, Nov. 1994), pp. 13–23.

[30] YAO, F., DEMERS, A. J., AND SHENKER, S. A scheduling model for reduced CPU energy. In *Proc. 36th Annual IEEE Symposium on the Foundations of Computer Science* (Milwaukee, Wisconsin, Oct. 1995), pp. 374–382.