Optimal Sleep/Wake Scheduling for Time-Synchronized Sensor Networks with QoS Guarantees

Yan Wu, Sonia Fahmy, Ness B. Shroff
Center for Wireless Systems and Applications (CWSA), Purdue University
E-mail: {wu26, fahmy}@cs.purdue.edu, shroff@ecn.purdue.edu

***Abstract*— We study sleep/wake scheduling for low-duty cycle sensor networks.[1] Our work is different from previous work in that we explicitly consider the effect of the synchronization error. We study a widely used synchronization scheme and show that the synchronization error is non-negligible, and using a conservative guard time is energy wasteful. Hence, we formulate an optimization problem to minimize the expected energy consumption, with the constraint that the message capture probability should be no less than a *threshold*. We find that the problem is non-convex, hence cannot be solved by conventional convex optimization techniques. By investigating the unique structure of the problem, we transform the problem into a convex equivalent, and solve it using an efficient search method. Simulations show that our scheme significantly outperforms schemes that do not intelligently consider the synchronization error. We also remove the assumption that the capture probability threshold is given, and study how to decide it to meet the Quality of Services (QoS) requirements of the application.**

I. Introduction

An important class of sensor network applications is applications employing a large number of sensor nodes, where each sensor *periodically* produces a small amount of data and reports to a single (or a few) base station(s). This application class includes many typical sensor network applications such as habitat monitoring [1], [2], civil structure monitoring [3], and factory maintenance [4]. Several practical systems have been developed for these applications.

To manage the large number of sensor nodes in such applications, a scalable method is to periodically group sensors within a geographical region into a cluster. The sensors can be managed locally by a cluster head (CH) – a node elected to coordinate the nodes within the cluster and to be responsible for communication between the cluster and the base station or other cluster heads. Clustering provides a convenient framework for resource management, data fusion, and local decision making [5]–[7]. One problem with clustering is that the cluster head is heavily utilized for both intra-cluster coordination and inter-cluster communications. Therefore, the cluster head will quickly deplete its energy. To address this concern, periodic re-clustering is performed to distribute the energy consumption among sensor nodes.

[1]This research has been sponsored in part by NSF grants 0238294 and 0207728, an Indiana 21st century grant, and a Tellabs foundation fellowship.

Sleep/wake scheduling has also been proposed to reduce energy consumption in sensor networks. The basic idea is to put the radio to sleep during idle times and wake it up right before message transmission/reception. This requires fine-grained synchronization between the sender and the receiver, so that they can wake up at the same time to communicate with each other. Prior work on sleep/wake scheduling assumes that the underlying synchronization protocol can provide nearly perfect (e.g., micro-second level) synchronization, or assumes an upper bound on the clock disagreement, and uses it as a guard time to compensate for the synchronization error. The awake period is lengthened by the guard time to combat synchronization errors. In practice, due to non-deterministic errors in time synchronization, as time progresses, clock disagreement becomes more and more significant. Periodic re-synchronization can prevent the clocks from drifting, but for low duty cycle sensor networks, frequent re-synchronization would consume a significant amount of energy compared to communication/sensing. Using an upper bound on clock disagreement as guard time will also significantly waste energy, since the synchronization error is non-deterministic.

In this work, we study the sleep/wake scheduling problem in clustered low duty cycle sensor networks. The nodes in the cluster are assumed to continuously monitor their environment and *periodically* report to the cluster head. Because the traffic is highly regular and the load is very low, the cluster head can go to sleep when no activity is present, and only wake up intermittently to send and receive messages. The following question hence becomes critical: *When should the cluster head wake up and how long should it stay active?*

With perfect synchronization, the cluster head and the cluster member simply agree upon a time and wake up simultaneously. We investigate a widely used synchronization scheme, proposed in the well-known Reference Broadcast Synchronization protocol [8]. We find that this scheme, although it achieves precise synchronization *immediately after* the exchange of synchronization messages, *has non-negligible clock disagreement as time progresses*. This, in fact, is true for most synchronization schemes, i.e., due to non-deterministic factors, the synchronization error will grow with time until the next exchange of synchronization messages. We conclude that the design of an effective sleep/wake scheduling algorithm

1-4244-0476-2/06/$20.00 ©2006 IEEE.

must take into account the impact of synchronization error, and study the optimal sleep/wake scheduling scheme with consideration of the synchronization error. We show that there is an inherent trade-off between energy consumption and message delivery performance (defined as the message capture probability in this work). In order to reduce energy consumption but still guarantee high message delivery performance, we formulate an optimization problem to minimize the expected energy consumption, with the constraint that the message capture probability should be no less than a *threshold*. The optimization problem is non-convex. However, by investigating the unique structure of the problem, we are able to transform it into a convex equivalent, and solve it using an efficient search method.

The remainder of this paper is organized as follows. Section II reviews related work. Section III describes the system model and formulates the problem. Section IV gives our sleep/wake scheduling algorithm and discusses implementation considerations. Section V presents the simulation results. Section VI studies how to assign the message capture probability threshold for messages from each individual node to meet the Quality of Services (QoS) requirements of the application. Section VII concludes the paper.

II. Background and Related Work

We first discuss time synchronization, then review previous work on sleep/wake scheduling.

A. Time Synchronization for Sensor Networks

Time synchronization has been studied in the context of wireless sensor networks [8]–[16]. Clock disagreement among sensor nodes is essentially due to two effects: *phase offset* and *clock skew*. Phase offset corresponds to clock disagreement between nodes at a given instant. Clock skew is because the crystal oscillators used on sensor nodes are imperfect, i.e., there is a difference between the expected frequency and the actual frequency. Further, the frequency may be time varying due to environmental factors, including variations in temperature and pressure [17]. Due to clock skew, clocks diverge over time.

Several synchronization protocols have been proposed to estimate the phase offset and clock skew. Elson et al. [8] proposed a receiver-receiver synchronization scheme called Reference-Broadcast Synchronization (RBS). In RBS, a node sends beacons to its neighbors using physical-layer broadcast. The recipients use the arrival time of the broadcast as a reference point to compare their times. RBS removes the non-determinism in the transmission time, channel access time, and propagation delay. The only non-determinism is in the packet reception time. To estimate the clock skew and phase offset, least square linear regression is used. Ganeriwal et al. [10] propose a sender-receiver synchronization approach called Timing-sync Protocol for Sensor Networks (TPSN). The TPSN approach time-stamps synchronization messages at the Medium Access Control (MAC) layer. It eliminates errors caused by access time and propagation delay via two way message exchange. A shortcoming of TPSN is that it does not estimate the clock skew of the nodes. Maroti et al. [13] combine the MAC layer time-stamping of TPSN with clock skew estimation using linear regression, and demonstrate improved performance. In both RBS and TPSN, measurements show that the synchronization error follows a well-behaved *normal distribution with zero mean*. We will use this observation to model the error distribution in our work.

Both RBS and TPSN give high-precision synchronization, i.e., the clock disagreement *immediately after* the exchange of synchronization messages is on the order of several tens of microseconds. However, due to the estimation error in the clock skew, the clock disagreement becomes *more significant* as time progresses.

B. Sleep/Wake Scheduling for Sensor Networks

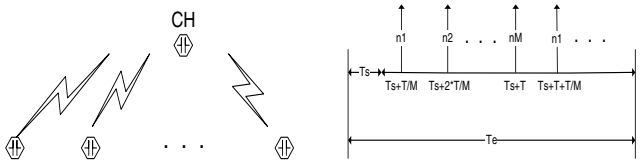
Sleep/wake scheduling for sensor networks has been extensively investigated [18]–[25]. The basic idea is to put the radio to sleep during idle times, and wake it up right before message transmission/reception. This requires fine-grained synchronization between the sender and the receiver, so that they can wake up at the same time to communicate with each other. Existing sleep/wake scheduling schemes either assume that the underlying synchronization protocol can provide nearly perfect (e.g., micro-second level) synchronization, or assume an upper bound on the clock disagreement, and use it as a guard time to compensate for the synchronization error. However, as pointed out in Section II-A, existing synchronization protocols like RBS or TPSN achieve micro-second level synchronization at the time instant *immediately following* the exchange of synchronization messages. But due to estimation errors in the clock skew, the clocks will gradually drift as time progresses, until the next exchange of synchronization messages. To see how significant the clock disagreement can be, consider two nodes that have agreed to rendezvous on the radio channel once every 100 seconds to exchange a 20-byte message. Using a 19.2 kbps radio such as RF Monolithics [26], 20 bytes can be transmitted in about 8 ms. The radio must be awakened early to account for clock disagreement. Let the estimation error of the clock skew be 10 ppm[2], i.e., the clocks of the two nodes drift away from each other 10 μs each second. After 100 seconds, the clocks will drift by $10\ \mu s \times 100 = 1\ ms$, which is non-negligible compared to the actual message transmission time. Ye and Heidemann [28] considered the effect of synchronization error in the design of a polling-based MAC protocol called Scheduled Channel Polling (SCP). In SCP, the receivers periodically poll the channel for network activity, and the sender uses a preamble to wake up the receiver before sending the actual message. To accommodate the clock disagreement, they extend the preamble by a guard time, which is equal to the product of the maximum clock skew and the time elapsed since the last synchronization. Using this worst case clock disagreement as the guard time can compensate for

[2]From the datasheet of Mica Motes [27], the clock skew with respect to the standard clock is up to 50 ppm, thus the relative clock skew between two sensor nodes can be 100 ppm in the worst case.

the synchronization error; however, energy efficiency can be further improved by exploiting the non-deterministic nature of the synchronization error.

III. System Model

We consider the example of a cluster which has been constructed using an existing clustering protocol (e.g., [5]–[7]). The cluster consists of a single cluster head (CH) and M cluster member nodes $n_1, n_2, \ldots n_M$ (Fig. 1(a)). Time is divided into recurring *epochs* with constant duration T_e. As with many MAC protocols for sensor networks [19]–[21], each epoch begins with a synchronization interval, T_s, followed by a transmission interval (Fig. 1(b)). During the synchronization interval, the cluster members synchronize with the CH, and no transmissions are allowed. During the transmission interval, each member continuously monitors its environment and sends one message to the CH every T seconds. Each transmission interval contains one or more rounds of transmissions, i.e., $T_e = T_s + NT, N \geq 1$. The transmissions from different members are equispaced, i.e., transmissions from node n_i and n_{i+1} are separated by $\frac{T}{M}$. Re-clustering of the network may occur at a lower frequency than synchronization, i.e., the time between re-clustering the network consists of one or more epochs.



(a) A cluster with a single head and multiple member nodes

(b) Equispaced upstream transmissions

Fig. 1. System model

We make the following assumptions about our system:

(1) Communication pattern: In this work, we focus on intra-cluster communications. Another important question is: after a CH receives messages from its members, how can it transmit messages to the base station (possibly via other cluster heads), and how can it wake up to relay messages from other cluster heads? This can be achieved, for example, by further dividing each transmission interval into two subintervals. One subinterval is for intra-cluster communications, and the other is for inter-cluster communications when the CH is always active. In the remainder of the paper, we only focus on intra-cluster communications. Our future work plans include considering the problem of energy-efficient sleep/wake scheduling for inter-cluster communications.

We further assume that neighboring clusters use orthogonal frequency channels and do not interfere with each other. This assumption is reasonable since the data rate requirements of sensor networks are usually low, typically around $10-40$ kbps. If we assume the radios operate in ISM-900 bands ($902-928$ MHz), then we have more than a thousand frequency bands to choose from.

(2) Clock skew: Vig [17] discussed the behavior of general off-the-shelf crystal oscillators. Because of imprecision in the manufacturing process and aging effects, the frequency of a crystal oscillator may be different from its desirable value. The maximum clock skew is usually specified by the manufacturer and is no larger than 100 ppm. The clock skew, however, can change due to environmental factors, such as variations in temperature, pressure, voltage, radiation, and magnetic fields. Among these environmental factors, temperature has the most significant effect. For general off-the-shelf crystal oscillators, when temperature significantly changes, the variation in the clock skew can be up to several tens of ppm, while the variation caused by other factors is far below 1 ppm. Observe, however, that temperature does not change dramatically within a few minutes in typical sensor environments. If the epoch duration T_e is chosen according to the temperature change properties of the environment, we can assume that the clock skew for each node is constant over each epoch. This is consistent with the observations in [16].

The crystal oscillator used by Mica Motes [27] is one type of off-the-shelf crystal oscillator, with similar characteristics to those discussed above. Specifically, its maximum clock skew can be up to 50 ppm.

(3) Radio hardware: For the transmitter circuit, we assume that the sender can precisely control when the message is sent out onto the channel using its *own* clock. This is consistent with the measurements in [10]. Therein, it is observed that non-determinism at the sender is negligible compared to non-determinism at the receiver, i.e., there are minor random effects at the sender.

For the receiver circuit, we assume that if there is an incoming message, the receiver circuit can detect the signal immediately. This is a close approximation of the real situation, since modern transceivers can detect the incoming signal within several microseconds [29]. We further assume that once the receiver circuit detects an incoming message, it can let the processor know, so that the processor will keep the radio active until the reception is completed. This can be easily achieved using a 1-bit status register.

(4) Collisions: We assume that the separation between transmissions from different members, namely, $\frac{T}{M}$, is large enough so that the collision probability between transmissions from different members is negligible. This assumption is reasonable for low duty cycle sensor networks. Consider a large cluster of $M = 100$ members and each member transmits to the CH every $T = 60$ seconds. The separation is $\frac{T}{M} = 600\ ms$, which is much larger than the message transmission time in sensor networks. In addition, the cluster nodes will be re-synchronized before the clock disagreement becomes large enough to cause significant collision probability.

(5) Energy expenditure: Measurements show that among all the sensor node components, the radio consumes the most significant amount of energy. In Section IV, we will show that the computational complexity of our scheduling algorithm is

very low. Therefore, in this work, we only account for energy consumption of the radio.

(6) Propagation delay: The communication range for sensor nodes is very short, typically < 100 meters. Therefore, we consider the propagation delay to be negligible.

A. Synchronization Algorithm

We adopt a *widely used* synchronization scheme, and study the sleep/wake scheduling problem under this scheme[3]. The scheme was first proposed in RBS [8], and was later adopted by several protocols and system implementations [11]–[15]. The scheme includes two steps: (1) Exchange synchronization messages to obtain multiple pairs of corresponding time instants; (2) Use linear regression to estimate the clock skew and phase offset.

Either a receiver-receiver approach or a sender-receiver approach can be used in the synchronization protocol. For the purpose of intra-cluster communication, the members only need to synchronize locally with the CH. Thus, at the start of each epoch j, each cluster member n_i will exchange several synchronization messages with the CH and obtain N_s pairs of corresponding time instants $(C(j,k), t_i(j,k)), k = 1 \ldots N_s$, where $C(j,k), t_i(j,k)$ denote the k^{th} time instant of the CH and of node n_i in epoch j respectively.

Under the assumption that the clock skew of each node does not change over the epoch, the clock time t_i of node n_i during an epoch is a linear function of the CH clock time C, i.e., $t_i(C) = a_i(j)C + b_i(j)$, where $a_i(j), b_i(j)$ denote the relative clock skew and phase offset (respectively) between n_i and CH in epoch j.

Due to the non-determinism in the synchronization protocols, the time correspondence obtained via exchange of synchronization messages is not exactly accurate and contains error, i.e.,

$$t_i(j,k) = a_i(j)C(j,k) + b_i(j) + e_i(j,k), \qquad (1)$$

where $e_i(j,k)$ is the random error caused by non-determinism in the system. Measurements show that $e_i(j,k)$ follows a normal distribution with zero mean $N(0, \sigma_0^2)$, and σ_0 is on the order of several tens of microseconds. Specifically, the chi-square test shows a 99.8% confidence [8], which strongly indicates the validity of this model.

In each epoch j, pairs $(C(j,k), t_i(j,k)), k = 1 \ldots N_s$ are obtained during the synchronization interval. Then, linear regression is performed on these N_s pairs to obtain estimates of $a_i(j), b_i(j)$, denoted by $\hat{a}_i(j), \hat{b}_i(j)$.

B. Problem Definition

Suppose that during epoch j, node n_i has a packet (message) p to send at CH clock time τ_p, where $jT_e \leq \tau_p \leq (j+1)T_e$. The node first translates τ_p into its own time using the estimates $(\hat{a}_i(j), \hat{b}_i(j))$, i.e., $\hat{t}_i(\tau_p) = \hat{a}_i(j)\tau_p + \hat{b}_i(j)$. Then, it sends out the message at $\hat{t}_i(\tau_p)$ according to its own clock.

[3]We select this scheme for illustration purposes, but our sleep/wake scheduling solution works with most synchronization schemes.

The CH clock time corresponding to $\hat{t}_i(\tau_p)$ is:

$$\hat{\tau}_p = \frac{\hat{t}_i(\tau_p) - b_i(j)}{a_i(j)} = \tau_p + \frac{(\hat{a}_i(j) - a_i(j))\tau_p + (\hat{b}_i(j) - b_i(j))}{a_i(j)}. \qquad (2)$$

If the estimation is exact, i.e., $(\hat{a}_i(j), \hat{b}_i(j)) = (a_i(j), b_i(j))$, then from Equation (2), $\hat{\tau}_p = \tau_p$, i.e., n_i will transmit precisely at τ_p. Under our assumption of negligible propagation delay, τ_p is equal to the time at which p *should* arrive if the synchronization is perfect, i.e., the *scheduled arrival time*, while $\hat{\tau}_p$ is equal to the time that p *actually* arrives, i.e., the *actual arrival time*. Hence, $\hat{\tau}_p = \tau_p$ means that the actual arrival time is exactly the same as the scheduled arrival time. If this is true, the CH simply wakes up at τ_p to receive the message.

However, as given in Equation (1), random errors exist in the measurements. Therefore, $(\hat{a}_i(j), \hat{b}_i(j))$ is also random. As a result, the actual arrival time $\hat{\tau}_p$ will deviate from the scheduled arrival time τ_p. To compensate for this random deviation and to "capture" (receive) the message, the CH needs to wake up earlier than τ_p and stay active for some time (Fig. 2). This leads to the following question: *When should the cluster head wake up and how long should it stay active?*

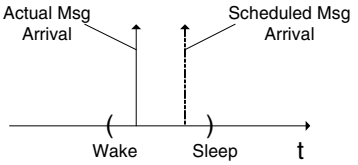


Fig. 2. Wake up interval to capture the message

Intuitively, if the CH wakes up much earlier than τ_p and stays active for a long time, it has a high probability of "capturing" the message; however, waking up early and staying active for a long time wastes energy. In order to reduce energy consumption, yet still guarantee high message delivery performance, we formulate the following optimal sleep/wake scheduling problem which attempts to minimize the expected energy consumption with constraints on the "capture" probability.

Let p be a message from n_i to arrive during epoch j, i.e., scheduled arrival time $\tau_p \in (jT_e, jT_e + T_e)$. Let $\hat{\tau}_p$ be the actual arrival time at which p arrives at the CH, as defined in Equation (2). To capture p, the CH wakes up at w_p. If the message does not arrive until s_p, the CH goes back to sleep at s_p; if the message arrives between w_p and s_p, the CH remains active until the message is received, which may be earlier or later than s_p depending upon the actual arrival time and message length. Our goal is to determine w_p and s_p to minimize the expected energy consumption as described by the following optimization problem:

(A) Minimize $E = (s_p - w_p)\alpha_I Prob\{\hat{\tau}_p \notin (w_p, s_p)\} + \int_{w_p}^{s_p} \{(x - w_p)\alpha_I + \frac{L_p}{R}\alpha_r\} f_{\hat{\tau}_p}(x)dx$

such that $Prob\{\hat{\tau}_p \in (w_p, s_p)\} \geq th$,

where:

- α_I/α_r is the power consumption during idle/reception time;
- L_p is the length of the message;
- R is the data rate;
- $f_{\hat{\tau}_p}()$ is the Probability Density Function (PDF) of $\hat{\tau}_p$;
- th is the threshold on the capture probability, $0 < th < 1$. Its value should be decided by the QoS requirements of the application. In this work, we assume that the value of th is *already given* and is the same for messages from different cluster members.

In problem (A), the first term corresponds to the expected energy consumption when the message is missed, i.e., $\hat{\tau}_p \notin (w_p, s_p)$. In this case, the CH stays active during the time interval (w_p, s_p) and consumes $(s_p - w_p)\alpha_I$ amount of idle energy. The second term corresponds to the expected energy consumption when the message is received. Suppose the message arrives at $x \in (w_p, s_p)$, then, in addition to the reception energy, the CH will consume $(x - w_p)\alpha_I$ amount of idle energy, i.e., the energy needed to remain idle for (w_p, x).

IV. Solution

We first compute the PDF $f_{\hat{\tau}_p}(x)$, transform the problem, and then solve the equivalent formulation.

By linear regression analysis [30], we find that $\hat{\tau}_p$ is *normally* distributed and

$$E(\hat{\tau}_p) = \tau_p, \tag{3}$$
$$\sigma_p^2 \equiv VAR(\hat{\tau}_p) = \frac{\sigma_0^2}{a_{ij}^2}[\frac{1}{N_s} + \frac{1}{N_s}\frac{(\tau_p - \overline{C_{jk}})^2}{\overline{C_{jk}^2} - (\overline{C_{jk}})^2}].$$

where $\overline{C(j,k)} = \frac{\sum_{k=1}^{N_s} C(j,k)}{N_s}, \overline{C^2(j,k)} = \frac{\sum_{k=1}^{N_s} C^2(j,k)}{N_s}$.

Substituting Equation (3) into problem (A), and letting $\hat{\tau} = \frac{\hat{\tau}_p - \tau_p}{\sigma_p}$, then $w = \frac{w_p - \tau_p}{\sigma_p}, s = \frac{s_p - \tau_p}{\sigma_p}$, i.e., $\hat{\tau}, (w, s)$ are the normalized arrival time and normalized wake up interval respectively. With simple algebraic operations, problem (A) becomes:

(A1) Minimize $F(w,s) = (s - w)\sigma_p\alpha_I - [Q(w) - Q(s)]s\sigma_p\alpha_I + [g(w) - g(s)]\sigma_p\alpha_I + [Q(w) - Q(s)]\frac{L_p}{R}\alpha_r$,
such that $Q(w) - Q(s) \geq th$,

where $g(x)$ is the probability density function for the standard normal distribution, and $Q(x)$ is the complementary cumulative distribution function.

The main difficulty in solving (A1) is that the problem is not a convex optimization problem, which can be easily shown by computing the Hessian matrix. Due to the non-convexity, we cannot use conventional convex optimization techniques [31] to find the optimal solution. Hence, we look into the structure of problem (A1) and show that it has certain unique properties that enable us to transform it into a convex equivalent, and solve the equivalent using an efficient search method.

We start by showing the following lemma.

Lemma 1: $\frac{\partial F}{\partial s} > 0$

Proof: We compute $\frac{\partial F}{\partial s}$ as follows.

$$\frac{\partial F}{\partial s} = [1 - Q(w) + Q(s)]\sigma_p\alpha_I + Q^{'}(s)(s - w)\sigma_p\alpha_I + Q^{'}(s)w\sigma_p\alpha_I + \frac{1}{\sqrt{2\pi}}se^{-\frac{s^2}{2}}\sigma_p\alpha_I + (-Q^{'}(s))\frac{L_p}{R}\alpha_r.$$

Since $Q^{'}(s) = -\frac{1}{\sqrt{2\pi}}e^{-\frac{s^2}{2}}$, put it in and we get

$$\frac{\partial F}{\partial s} = [1 - Q(w) + Q(s)]\sigma_p\alpha_I + \frac{1}{\sqrt{2\pi}}e^{-\frac{s^2}{2}}\frac{L_p}{R}\alpha_r.$$

Since $\forall x \in R, 0 \leq Q(x) \leq 1$, therefore $1 - Q(w) + Q(s) \geq 0$. Consequently,

$$\frac{\partial F}{\partial s} = [1 - Q(w) + Q(s)]\sigma_p\alpha_I + \frac{1}{\sqrt{2\pi}}e^{-\frac{s^2}{2}}\frac{L_p}{R}\alpha_r > 0. \quad \blacksquare$$

Here is an intuitive explanation of $\frac{\partial F}{\partial s} > 0$. As in (B), we write

$$F(w,s) = \sigma_p\alpha_I(s - w)Prob\{\hat{\tau} \notin (w,s)\} + \int_w^s (x - w)\sigma_p\alpha_I g(x)dx + \int_w^s \frac{L_p}{R}\alpha_r g(x)dx.$$

We note that the first two terms correspond to the expected idle energy consumption, while the third term corresponds to the expected energy used to receive the message. Suppose the normalized wake up interval is changed from (w, s) to $(w, s + \Delta)$, we observe that:

- The expected energy for receiving the message increases because the capture probability is larger;
- The change in the idle energy consumption is illustrated in Fig. 3. In the figure, t_1, t_2, and t_3 are three possible message arrivals, where $t_1 \in [w, s], t_2 \in (s, s + \Delta), t_3 \notin [w, s + \Delta)$. $I_i^{'}, I_i$ $i = 1, 2, 3$ are the idle time for the message arrival at t_i before and after s is increased to $s + \Delta$, respectively.
  - If the message arrival is in $[w, s]$, e.g., t_1, the idle energy consumption does not change;
  - If the message arrival is in $(s, s + \Delta)$, e.g., t_2, the idle energy increases;
  - If the message arrival is at another time, e.g. t_3, the idle energy increases.

Therefore, as the normalized wake up interval changes from (w, s) to $(w, s + \Delta)$, the idle energy does not decrease, while the expected receiving energy always increases. This explains why the total energy consumption increases with s.

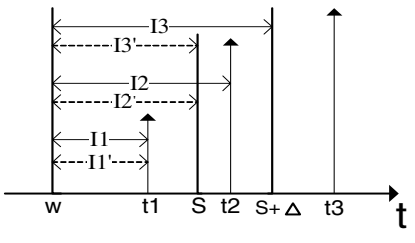


Fig. 3. Changes in the idle time when s increases

The next proposition shows that the optimal solution always appears at the boundary of the region $Q(w) - Q(s) \geq th$.

Proposition 1: Let (w^*, s^*) be the optimal solution to (B), then $Q(w^*) - Q(s^*) = th$.

Proof: We prove this by contradiction. Suppose $Q(w^*) - Q(s^*) > th$.

Because $Q(x)$ is continuous, $\exists r_1 > 0$ s. t.

$$Q(w^*) - Q(s) \geq th, \forall s^* - r_1 \leq s \leq s^* + r_1.$$

Meanwhile, we have

$$F(w^*, s^* - \Delta) - F(w^*, s^*) = -\frac{\partial}{\partial s} F(w^*, s^*)\Delta + O(\Delta^2).$$

As shown in Lemma 1, $\frac{\partial}{\partial s} F(w^*, s^*) > 0$, so $\exists r_2 > 0$ s. t.

$$F(w^*, s^* - \Delta) - F(w^*, s^*) < 0, \forall 0 < \Delta \leq r_2.$$

Pick $r = \min(r_1, r_2)$, then it satisfies $Q(w^*) - Q(s^* - r) \geq th$ and $F(w^*, s^* - r) - F(w^*, s^*) < 0$, which means that $(w^*, s^* - r)$ is a feasible point and $F(w^*, s^* - r)$ is lower than the minimum. This is contradictory to the fact that (w^*, s^*) is the optimal solution. ■

The physical meaning of $Q(w^*) - Q(s^*) = th$ is that under the optimal scheduling policy, the capture probability is always equal to the threshold th. This can be easily understood from Lemma 1. If $Q(w) - Q(s) > th$, then we reduce s by a small amount (go to sleep earlier by) Δ. From Lemma 1, the total energy consumption decreases, yet the capture probability still exceeds the threshold. Thus, $(w, s - \Delta)$ is a better solution than (w, s). Hence, the optimal solution must satisfy $Q(w^*) - Q(s^*) = th$.

Substituting $Q(w^*) - Q(s^*) = th$, formulation (A1) becomes:

(A2) Minimize $F(w, s) = [(1 - th)s - w + g(w) - g(s)]\sigma_p \alpha_I + th\frac{L_p}{R}\alpha_r$,
such that $Q(w) - Q(s) = th$.

We further simplify the formulation as follows. First, because $th\frac{L_p}{R}\alpha_r$ does not depend on w and s, we remove it from $F(w, s)$. Second, all the remaining terms of $F(w, s)$ have $\sigma_p \alpha_I$, so we can extract $\sigma_p \alpha_I$. Finally, we express s as a function of w, $s(w) = Q^{-1}(Q(w) - th), w < Q^{-1}(th)$. Now, the formulation becomes:

(A3) Minimize $G(w) = (1 - th)s(w) - w + g(w) - g(s(w))$,
such that $s(w) = Q^{-1}(Q(w) - th), w < Q^{-1}(th)$.

So far, we have transformed the original formulation (A) into an equivalent formulation (A3). Next, we solve (A3).

We first show that $G(w)$ is a convex function of w.

Proposition 2: $G^{''}(w) > 0$.

Proof: We will use the following properties of $g(x)$ in the proof: $g(x) = g(-x), g^{'}(x) = -xg(x)$.

From formulation (C),

$$G^{''} = (1 - th)s^{''}(w) + (s - w)g^{'}(w) + [s^{'}(w) - 1]g(w).$$

It is sufficient to show that

$$s^{''}(w) > 0$$

and

$$(s - w)g^{'}(w) + [s^{'}(w) - 1]g(w) \geq 0.$$

We first examine $s^{''}(w)$. Taking the derivative on both sides of $Q(w) - Q(s(w)) = th$, we have

$$-g(w) + g(s)s^{'}(w) = 0 \Longrightarrow s^{'}(w) = \frac{g(w)}{g(s)}.$$

Thus,

$$s^{''}(w) = \frac{g^{'}(w)g(s) - g(w)g^{'}(s)s^{'}(w)}{g^2(s)}.$$

Since $g^{'}(x) = -xg(x)$, we have

$$s^{''}(w) = \frac{-wg(w)g^2(s) + sg(s)g^2(w)}{g^3(s)} = \frac{g^2(w)}{g(s)}[\frac{s}{g(s)} - \frac{w}{g(w)}].$$

Since $(\frac{x}{g(x)})^{'} = \frac{g(x) + x^2 g(x)}{g^2(x)} > 0$, $\frac{x}{g(x)}$ is a strictly increasing function. Therefore,

$$s > w \Longrightarrow \frac{s}{g(s)} - \frac{w}{g(w)} > 0 \Longrightarrow s^{''}(w) > 0.$$

Next,

$$(s - w)g^{'}(w) + [s^{'}(w) - 1]g(w) = -w(s - w)g(w) + \frac{g(w)}{g(s)}[g(w) - g(s)] = g(w)[-w(s - w) + \frac{g(w) - g(s)}{g(s)}].$$

As $g(w) > 0$, it is sufficient to show that

$$-w(s - w) + \frac{g(w) - g(s)}{g(s)} > 0.$$

There are three cases:

- $w < s \leq 0$
  By the Mean Value Theorem,
  $$g(w) - g(s) = (w - s)g^{'}(\zeta), \zeta \in [w, s],$$
  we have
  $$-w(s - w) + \frac{g(w) - g(s)}{g(s)} = -w(s - w)[1 - \frac{g(\zeta)}{g(s)}\frac{\zeta}{w}].$$
  Then
  $$\begin{aligned} w \leq \zeta \leq s \leq 0 \quad &\Longrightarrow \quad 0 \leq \frac{g(\zeta)}{g(s)} \leq 1, 0 \leq \frac{\zeta}{w} \leq 1 \\ &\Longrightarrow \quad 1 - \frac{g(\zeta)}{g(s)}\frac{\zeta}{w} \geq 0 \\ &\Longrightarrow \quad -w(s - w)[1 - \frac{g(\zeta)}{g(s)}\frac{\zeta}{w}] \geq 0. \end{aligned}$$
- $0 \leq w < s$
  This case can be proved using the Mean Value Theorem as above: observe that $0 \leq w < s$ implies $\frac{g(\zeta)}{g(s)} \geq 1$ and $\frac{\zeta}{w} \geq 1, \forall \zeta \in [w, s]$.
- $w \leq 0 \leq s$
  If $g(w) \geq g(s)$, then
  $$-w(s - w) \geq 0, \frac{g(w) - g(s)}{g(s)} \geq 0 \Longrightarrow -w(s - w) + \frac{g(w) - g(s)}{g(s)} \geq 0.$$

Otherwise, $g(w) < g(s) \Longrightarrow w < -s$. Hence, by the Mean Value Theorem,

$$-w(s-w)+\frac{g(w)-g(s)}{g(s)} = -w(s-w)+$$
$$+\frac{g(w)-g(-s)}{g(-s)} = -w(s-w)-\frac{\zeta g(\zeta)}{g(-s)}(w+s)$$
$$= -w(s-w)[1+\frac{\zeta}{w}\frac{g(\zeta)}{g(-s)}\frac{w+s}{s-w}],$$

where $\zeta \in [w,-s]$.

$$w \le \zeta \le -s \le 0 \Longrightarrow 0 \le \frac{\zeta}{w} \le 1, 0 \le \frac{g(\zeta)}{g(-s)} \le$$
$$\le 1, -1 \le \frac{w+s}{s-w} \le 0 \Longrightarrow \frac{\zeta}{w}\frac{g(\zeta)}{g(-s)}\frac{w+s}{s-w} \ge -1$$
$$\Longrightarrow -w(s-w)[1+\frac{\zeta}{w}\frac{g(\zeta)}{g(-s)}\frac{w+s}{s-w}] \ge 0.$$

In all, $-w(s-w)+\frac{g(w)-g(s)}{g(s)} \ge 0$. Combining this with $s^{''}(w) > 0$, we have $G^{''}(w) > 0$. ■

Since $G(w)$ is convex, and the region $w \in (-\infty, Q^{-1}(th))$ is a convex region, then the local minimum is in fact a global minimum. The next proposition gives the position of the global minimum.

Proposition 3: Let w_0 be the global minimum, $w_l = Q^{-1}(\frac{1+th}{2}), w_u = min(0, Q^{-1}(th))$, then $w_0 \in (w_l, w_u)$, and is the unique minimum on this interval.

We give the proof in [32]. Since w_0 is the unique minimum on (w_l, w_u), we can use the Golden Search method to find w_0 [33]. The Golden Search method has logarithmic complexity of $O(log(\frac{1}{\delta}))$, where δ is the required precision. Hence, it can be efficiently implemented.

A. Example Implementation

We now describe an example implementation of our approach.

Cluster Initialization. After the clusters are established, the cluster head (CH) broadcasts the epoch duration T_e, synchronization interval T_s, and message frequency T to the cluster members, and lets the members know when they should transmit (according to the CH clock). In the following discussion, we will assume that the parameters and the transmission schedules for the members will not change in the system. In cases when these need to be changed, the CH simply makes the change and informs the members.

Synchronization. As we indicated in Section III-A, the synchronization scheme can utilize either a receiver-receiver approach (RBS) or a sender-receiver approach (TPSN). Our example implementation uses RBS. We first review how RBS works. In RBS, when two nodes A and B want to synchronize with each other, they need a separate beacon node. The beacon node broadcasts a beacon, which is received at T1 and T2 by A and B respectively. Specifically, let the relationship between node A's clock and node B's clock be $t_B = at_A + b$. Then, $T2 = aT1 + b + e$, where e is the non-deterministic factor, which follows a normal distribution $N(0, \sigma_0^2)$. Hence, one pair of corresponding times $(T1, T2)$ is obtained. Additional pairs can be obtained using multiple broadcast beacons.

To use RBS in the cluster, the CH selects a member as the beacon node. This member sends reference beacons using a sufficiently high power level[4] for the beacons to be received by all other members and the CH. The cluster members then exchange the arrival times of the beacons with the CH and obtain multiple pairs of corresponding time instants. The cluster members will use these pairs to estimate $(a_i(j), b_i(j))$ as described in Section III-A. At this stage, all members will have synchronized with the CH except the beacon node. The CH then selects another member to send reference beacons, so that the original beacon node can synchronize with the CH.

To further conserve energy, the cluster members and the CH do not always stay active during the synchronization interval. Instead, they go to sleep if there are no beacons and wake up right before beacons arrive. Observe, however, that because of clock disagreement, the CH and the cluster members may not rendezvous with the beacon node. This leads to the loopback problem: the synchronization message itself cannot be successfully exchanged because of synchronization error. To solve this problem, the cluster members and the CH use a guard time to compensate for synchronization error. This guard time is chosen to be $3\times$ message transmission time, while the clock disagreement is controlled such that it cannot go far beyond the message transmission time with high probability. Using this mechanism, in all our simulations, beacons and subsequent messages can be successfully communicated.

Determining the Wake up Schedule. To determine the wake up interval, the CH first computes (w^*, s^*) using the Golden Search method. The CH needs to do this computation only once. Next, the CH computes for each message p the value of σ_p using Equation (3). In Equation (3), σ_0 can be obtained from measurements which have already been taken, e.g., in RBS and TPSN. The difficulty is that we do not know $a_i(j)$. However, we can bound $a_i(j)$ in the following manner. According to [17], the maximum clock skew of most off-the-shelf crystal oscillators is no larger than 100 ppm (specifically for Mica Motes, the clock skew is no larger than 50 ppm and the bounds below still hold). Therefore, $\frac{1-10^{-4}}{1+10^{-4}} \le a_i(j) \le \frac{1+10^{-4}}{1-10^{-4}}$. Substituting into Equation (3), we have

$$0.9998^2 \le \sigma_p^2/[\sigma_0^2(\frac{1}{N_s}+\frac{1}{N_s}\frac{(\tau_p-\overline{C(j,k)})^2}{\overline{C^2(j,k)}-\overline{C(j,k)}^2})] \le 1.00021^2.$$

We choose $\sigma_p^2 = 1.00021^2\sigma_0^2[\frac{1}{N_s}+\frac{1}{N_s}\frac{(\tau_p-\overline{C(j,k)})^2}{\overline{C^2(j,k)}-\overline{C(j,k)}^2}]$. This value is no less than the actual σ_p, so the wake up interval will be larger than necessary and the capture probability will be slightly higher than th; yet the wake up interval is no

[4]We assume that each node has a fixed number of transmission power levels (as in Mica2 motes) and can transmit to the CH and all other cluster members using one of these power levels. This assumption is reasonable since in many clustering techniques, the transmission power level used by the members to communicate with the CH is much lower than the maximum. Therefore, a member node can increase the transmission power level to ensure its message can be received by the CH as well as other cluster members.

more than 0.04% larger than necessary, which causes little degradation in the energy consumption. After σ_p is obtained, the CH computes for each message p the wake up interval $(w_p, s_p) = (\tau_p + \sigma_p w^*, \tau_p + \sigma_p s^*)$.

V. Simulation Experiments

Our scheduling policy intelligently compensates for the synchronization error through dynamic computation of the wake up interval. We use the implementation described in Section IV-A. Another scheme that was previously used for compensating the synchronization error assumed an upper bound on the synchronization and use it as a *fixed* guard time. To evaluate the performance gain of dynamic adjustment of wake up intervals, we compare the performance of our scheme with that of the following fixed wake up interval scheme: The CH wakes up $\frac{L}{2}$ earlier than the scheduled message arrival time (recall from Section III-B that the scheduled arrival time is the time that the message *should* arrive). If the message does not arrive until $\frac{L}{2}$ after the scheduled arrival time, the CH goes back to sleep again; otherwise, it stays active until the message is received. To make the comparison fair, we use same message arrivals for both schemes.

We set the synchronization interval to 60 seconds. During each synchronization interval, the CH transmits to each cluster member in an equispaced manner, and obtains 2 pairs of corresponding times. We adopt the model used in RBS and TPSN to characterize the synchronization error. Specifically, the synchronization error is normally distributed with zero mean, $N(0, \sigma_0^2)$. In our simulations, we choose $\sigma_0 = 36.5\ \mu s$, which is derived from [10][5]. The clock skew of each node is chosen uniformly from $[1 - 50 \times 10^{-6}, 1 + 50 \times 10^{-6}]$ [27].

Table I summarizes the simulation parameters and other system constants. Unless otherwise specified, all the simulation results are averaged over 1000 runs.

TABLE I
SIMULATION PARAMETERS AND SYSTEM CONSTANTS

Threshold th	0.9
Idle power α_I (mW)	13
Receiving power α_r (mW)	13
Data rate R (kbps)	19.2
Message length L_p (byte)	8
Number of cluster member nodes M	10
Epoch duration T_e (minute)	20
Synchronization interval T_s (second)	60
Number of synchronization messages N_s	2
σ_0 (μs)	36.5
Transmission period T (second)	60

We first compare the message delivery performance of our scheme with the fixed interval scheme. From Equation (3), we have $VAR(\hat{\tau}_p) \equiv \sigma_p^2 = \frac{\sigma_0^2}{a_i^2(j)}[\frac{1}{N_s} + \frac{1}{N_s}\frac{(\tau_p - \overline{C(j,k)})^2}{C^2(j,k) - \overline{C(j,k)}^2}]$. Within an epoch, the variance of the actual arrival time increases with τ_p, the scheduled arrival time. This is because the clock drifts more and more as time progresses. As a result, for the fixed interval scheme, the capture rate will decrease as the scheduled arrival time increases. This is illustrated in Fig. 4(a). In the figure, we show how the capture rate changes as time goes on for both our scheme and the fixed interval scheme. We set $L = 3\ ms$ (recall that in the fixed interval scheme, the CH wakes up $\frac{L}{2}$ earlier than the scheduled arrival and stays active $\frac{L}{2}$ after the scheduled arrival). We observe that for the fixed interval scheme, the capture rate is very high at the beginning, but gradually decreases to below the threshold. If the message is scheduled to arrive near the end of the epoch ($T_e = 1200$), then the capture rate is only 0.55. In practice, this means that the fixed interval scheme cannot provide the threshold capture rate near the end of the epoch, which is undesirable. On the other hand, our scheme dynamically selects the wake up interval, so that the capture rate is always kept at no less than the threshold.

We now study the energy consumption properties of the two schemes. In Fig. 4(b), we vary L and compute the length of time (given as a percentage of the epoch) that the specified value of L is sufficient to give acceptable capture rates (above the threshold). For example, when $L = 3\ ms$, $percentage \approx 50\%$ means that if L is set to $3\ ms$, then for messages scheduled to arrive during the first half of the epoch, the capture rate is no less than the threshold; but if the message is scheduled to arrive during the second half of the epoch, the capture rate is lower than the threshold.

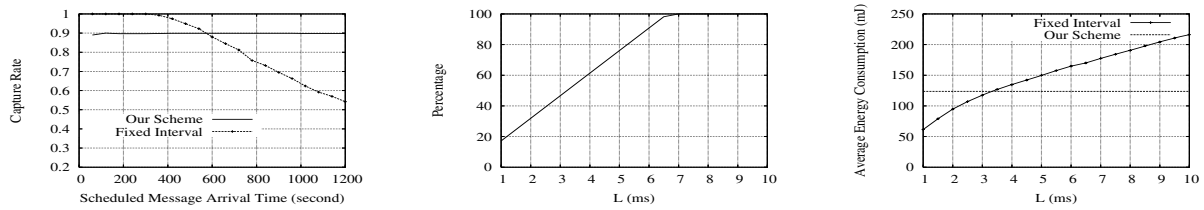
Fig. 4(c) illustrates the average cluster head energy consumption per epoch with different values of L. For comparison, we also include in the figure the average energy consumption per epoch of our scheme (the straight line). Since both our scheme and the fixed interval scheme use the same synchronization protocol, they consume the same amount of energy for synchronization. Therefore, we do not account for the energy consumed for synchronization here. From Fig. 4(a) and 4(b), we see that $L = 4\ ms$ can guarantee the threshold capture rate for only 60% of the epoch, but the energy consumption is already higher than our optimal scheduling scheme. In order to guarantee the threshold capture rate for the entire epoch, L must be set to at least 7 ms, with energy consumption 40% higher than our scheme.

We have also simulated the following combinations of parameters (please refer to Table I for the definition of these quantities): $th = 0.9, 0.8, 0.7, 0.6$; $L_p = 8, 16, 24$; $T_e = 20, 30, 40, 50, 60$; $T_s = 30, 60, 90, 120, 150$; $N_s = 2, 4, 6, 8, 10$; and $T = 60, 120, 180, 240$. The results were consistent with those discussed above, i.e., our scheme can guarantee a specified capture rate with lower energy consumption than the fixed interval scheme.

VI. Assignment of the Capture Probability Threshold for QoS Guarantees

Thus far, we have studied the optimal sleep/wake scheduling problem under the assumption that the capture probability threshold th is *already given*, and is identical for messages from different cluster members. An important question is:

[5] In [10], measurements show that the average absolute error is 29.1 μs. Therefore, $\int_{-\infty}^{\infty} |x| \frac{1}{\sqrt{2\pi}\sigma_0} e^{-\frac{x^2}{2{\sigma_0}^2}} dx = 29.1\ \mu s \Longrightarrow \sigma_0 \approx 36.5\ \mu s$.



(a) Comparison of the message delivery performance

(b) Message delivery performance of the fixed interval scheme

(c) Energy consumption of the two schemes

Fig. 4. Comparing with the fixed interval scheme

how to assign the capture probability threshold(s) to meet the QoS requirement of the application? In contrast to most Internet applications where different users compete for network resources, in many sensor networks, a group of sensors collaborate to perform common task(s). Therefore, the QoS for sensor networks is typically not decided by the performance of any individual node, but by the *collective* performance of all related nodes. For example, consider an environmental monitoring system where sensor nodes periodically report to a base station. Each message contains some sensing data and represents a certain amount of "information" of the environment. The base station uses the collected information to analyze interesting properties, e.g., the chemical contaminant in an area. The service quality is defined as the accuracy of the analysis, which is determined by the *total* amount of information collected from all the nodes.

In many sensor networks, heterogeneity may also exist among the sensor nodes. For example, some nodes may be equipped with an expensive sensor which provides high precision measurements, while others only have a low precision sensor for cost reasons. As a result, messages from different nodes may contain information of different qualities and represent different "values." To quantify the value of messages, Chen et al. [34] associate each message with a *utility value* to represent the amount of useful information contained in it. However, one assumption made in their work is that there is no redundancy in the network, and hence the data collected from different sensors contributes additive utilities. In reality, redundant sensors may be deployed in the sensing area, and the information collected by nearby sensors may be correlated[6]. Therefore, we associate each message with a *utility value*, which represents the amount of useful information contained in it; messages from the same node i have the same utility value U_i, $i = 1 \ldots M$. Unlike [34], we consider that messages from different nodes may be correlated and exhibit redundancy. Therefore, to guarantee the service quality, the CH only needs to collect a *certain proportion* of the total utility. As long as this proportion is obtained, the requirement on each individual node can be chosen with *flexibility*. In [32], we formulate an optimization problem for collecting this desired proportion of total utility with minimum energy consumption. Interested readers can refer to [32] for the solution and simulation results.

[6]Coverage scheduling can help reduce the redundancy, but experimental measurements [35] show that the correlation pattern can be very complex and it is hard to *completely* remove the redundancy.

VII. Conclusions and Future Work

In this paper, we have studied sleep/wake scheduling for low duty cycle sensor networks. Our work is distinguished from most previous work in that we explicitly consider the effect of synchronization error in the design of sleep/wake scheduling algorithm. In contrast, most previous work on sleep/wake scheduling either assumes perfect synchronization, or assumes an upper bound on the clock disagreement and uses a fixed guard time to compensate for the synchronization error. We utilized a widely used synchronization scheme, which was proposed in the well known RBS protocol. We demonstrated that this scheme, though it achieves microsecond level synchronization *immediately after* the exchange of synchronization messages, *turns out to have non-negligible clock disagreement as time progresses.* Therefore, the effect of synchronization error cannot be ignored in sleep/wake scheduling algorithms. Simply using a fixed guard time to compensate for the synchronization error is energy inefficient. In fact, we have shown that there is a trade-off between energy consumption and message delivery performance.

In order to reduce energy consumption, and yet guarantee high message delivery performance, we have formulated an optimization problem to minimize the expected energy consumption, with constraints on the message delivery performance. Due to the non-convexity of the problem, it cannot be directly solved using conventional convex optimization techniques. By exploiting the structure of the problem, we were able to transform the original non-convex problem into a convex equivalent, and solve it. Our scheme has a low computational burden, which is important for on-line implementation. Through simulations, we have compared the performance of our optimal sleep/wake scheduling scheme to a scheme without intelligent consideration of synchronization error. Simulation results illustrated that our scheme far outperforms such schemes. We have also removed the assumption that the capture probability threshold is *already given*, and studied how to decide it to meet the QoS requirement of the application.

Our work has largely been motivated by sensor networks running continuous monitoring applications. While this encompasses a large class of interesting applications, our approach is not limited to such applications. The major requirement of our approach is that the sender and receiver agree upon message arrival times. This requirement can also be satisfied in many sensor applications with a hybrid data delivery pattern [36], [37]. For example, a sensor network which monitors the concentration of a chemical can have two modes: the silent mode and the vigilant mode. When the average concentration collected from the entire network is below a certain threshold, the network is in silent mode, where nodes monitor the concentration and periodically report to the base station. When the base station finds the average concentration to exceed the threshold, it can trigger the network into vigilant mode, where transmissions can occur at any time. For such applications with a hybrid data delivery pattern, our approach is ideal for use in the silent mode.

We have focused on the single hop intra-cluster communication scenario in this work. After messages are received by the cluster head, they may need to be forwarded to the base station, potentially over multiple hops. The cluster head may also need to relay messages from other cluster heads to the base station. An important question is how to decide the sleep/wake schedules of the cluster heads over such multiple hops. Section III gave a simple example mechanism. A more efficient solution would be to develop an adaptive sleep/wake scheduling methodology, as in this paper, for inter-cluster communications. This is an open issue that merits further investigation.

REFERENCES

[1] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson, "Wireless Sensor Networks for Habitat Monitoring," in *Proc. of ACM WSNA*, September 2002.

[2] G. Tolle, J. Polastre, R. Szewczyk, N. Turner, K. Tu, S. Burgess, D. Gay, P. Buonadonna, W. Hong, T. Dawson, and D. Culler, "A Macroscope in the Redwoods ," in *Proc. of ACM SenSys*, 2005.

[3] S. C. Visweswara, A. A. Goel, and R. Dutta, "An Adaptive Ad-hoc Self-Organizing Scheduling for Quasi-Periodic Sensor Traffic," in *Proc. of IEEE SECON*, September 2004.

[4] K. Srinivasan, M. Ndoh, H. Nie, H. Xia, K. Kaluri, and D. Ingraham, "Wireless Technologies for Condition-Based Maintenance (CBM) in Petroleum Plants," *Proc. of the 1st International Conference on Distributed Computing in Sensor Systems (Poster Session)*, 2005.

[5] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "An Application-Specific Protocol Architecture for Wireless Microsensor Networks," *IEEE Transactions on Wireless Communications*, vol. 1, no. 4, pp. 660–670, October 2002.

[6] O. Younis and S. Fahmy, "Distributed Clustering in Ad-hoc Sensor Networks: A Hybrid, Energy-Efficient Approach," in *Proc. of IEEE INFOCOM*, Hong Kong, March 2004.

[7] S. Banerjee and S. Khuller, "A Clustering Scheme for Hierarchical Control in Multi-hop Wireless Networks," in *Proc. of IEEE INFOCOM*, April 2001.

[8] J. Elson, L. Girod, and D. Estrin, "Fine-Grained Network Time synchronization Using Reference Broadcasts," in *Proc. of USENIX/ACM OSDI*, 2002.

[9] J. Elson and K. Romer, "Wireless Sensor Networks: A new Regime for Time Synchronization," in *Proc. of HotNets-I*, October 2002.

[10] S. Ganeriwal, R. Kumar, and M. Srivastava, "Timing-sync Protocol for Sensor Networks," in *Proc. of ACM SenSys*, 2003.

[11] J. Elson, L. Girod, and D. Estrin, "Short Paper: A Wireless Time-Synchronized COTS Sensor Platform, Part I: System Architecture," in *Proc. of the IEEE CAS Workshop on Wireless Communications and Networking*, September 2002.

[12] Q. Gao, K. J. Blow, and D. J. Holding, "Simple algorithm for improving time synchronization in wireless sensor networks," *Electronics Letters*, vol. 40, pp. 889–891, July 2004.

[13] M. Maroti, B. Kusy, G. Simon, and A. Ledeczi, "The flooding time synchronization protocol," in *Proc. of ACM SenSys*, 2004.

[14] D. Cox, A. Milenkovic, and E. Jovanov, "Time Synchronization for ZigBee Networks," in *Proc. of the 37th Southeastern Symposium on System Theory*, Tuskegee, AL, March 2005.

[15] S. PalChaudhuri, A. K. Saha, and D. B. Johnson, "Adaptive clock synchronization in sensor networks," in *Proc. of IEEE/ACM IPSN*, 2004.

[16] S. Ganeriwal, D. Ganesan, H. Shim, V. Tsiatsis, and M. B. Srivastava, "Estimating Clock Uncertainty for Efficient Duty-Cycling in Sensor Networks," in *Proc. of ACM SenSys*, 2005.

[17] J. R. Vig, "Introduction to Quartz Frequency Standards," Technical Report SLCET-TR-92-1, Army Research Laboratory, October 1992, available at http://www.ieee-uffc.org/freqcontrol/quartz/vig/vigtoc.htm.

[18] S. Singh and C. Raghavendra, "PAMAS: Power Aware Multi-Access protocol with Signalling for Ad Hoc Networks," *ACM Computer Communication Review*, vol. 28, no. 3, pp. 5–26, July 1998.

[19] W. Ye, J. Heidenmann, and D. Estrin, "An Energy-Efficient MAC Protocol for Wireless Sensor Networks," in *Proc. of IEEE INFOCOM*, New York, NY, June 2002.

[20] T. Dam and K. Langendoen, "An Adaptive Energy-Efficient MAC Protocol for Wireless Sensor Networks," in *Proc. of ACM SenSys*, 2003.

[21] V. Rajendran, K. Obraczka, and J. Garcia-Luna-Aceves, "Energy-Efficient, Collision-Free Medium Access Control for Wireless Sensor Networks," in *Proc. of ACM SenSys*, 2003.

[22] J. Polastre, J. Hill, and D. Culler, "Versatile Low Power Media Access for Wireless Sensor Networks," in *Proc. of ACM SenSys*, 2004.

[23] G. Pei and C. Chien, "Low power TDMA in Large Wireless Sensor Networks," in *Proc. of IEEE MILCOM*, October 2001.

[24] S. Coleri, A. Puri, and P. Varaiya, "Power Efficient system for Sensor Networks," in *Proc. of IEEE ISCC*, July 2003.

[25] M. L. Sichitiu, "Cross-Layer Scheduling for Power Efficiency in Wireless Sensor Networks," in *Proc. of IEEE INFOCOM*, 2004.

[26] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. E. Culler, and K. S. J. Pister, "System Architecture Directions for Networked Sensors," in *Architectural Support for Programming Languages and Operating Systems*, 2000, pp. 93–104.

[27] http://www.tinyos.net/scoop/special/hardware.

[28] W. Ye and J. Heidemann, "Ultra-low duty cycle MAC with scheduled channel polling," USC/Information Sciences Institute, Tech. Rep. ISI-TR-2005-604, July 2005. [Online]. Available: http://www.isi.edu/∼johnh/PAPERS/Ye05a.html

[29] "CC1000 low power FSK transciever," Chipcon Corporation. http://www.chipcon.com.

[30] J. L. Devore, *Probability and Statistics for Engineering and the Sciences*. International Thomson Publishing Inc., 1995.

[31] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[32] Y. Wu, S. Fahmy, and N. B. Shroff, "Optimal QoS-aware Sleep/Wake Scheduling for Time-Synchronized Sensor Networks," Technical Report, 2006, available at http://www.cs.purdue.edu/homes/wu26/techrep.pdf.

[33] E. K. P. Chong and S. H. Zak, *An Introduction to Optimization*. John Wiley & Sons, Inc., 2001.

[34] W.-P. Chen and L. Sha, "An energy-aware data-centric generic utility based approach in wireless sensor networks," in *Proc. of IEEE/ACM IPSN*, 2004.

[35] F. Koushanfar, N. Taft, and M. Potkonjak, "Sleeping coordination for comprehensive sensing using isotonic regression and domatic partitions," in *Proc. of IEEE INFOCOM*, April 2006.

[36] Y. Yao and J. E. Gehrke, "Query processing in sensor networks," in *Proc. of the 1st Biennial Conference on Innovative Data Systems Research*, 2003.

[37] A. Boukerche, R. Werner, N. Pazzi, and R. Borges-Araujo, "A fast and reliable protocol for wireless sensor networks in critical conditions monitoring applications," in *Proc. of the 7th ACM MSWiM*, 2004.