Towards Optimal Sleep Scheduling in Sensor Networks for Rare-Event Detection

Qing Cao, Tarek Abdelzaher, Tian He, John Stankovic
Department of Computer Science, University of Virginia, Charlottesville, VA 22904
email:{qingcao, zaher, tianhe, stankovic}@cs.virginia.edu

Abstract— **Lifetime maximization is one key element in the design of sensor-network-based surveillance applications. We propose a protocol for node sleep scheduling that guarantees a bounded-delay sensing coverage while maximizing network lifetime. Our sleep scheduling ensures that coverage rotates such that each point in the environment is sensed within some finite interval of time, called the detection delay. The framework is optimized for rare event detection and allows favorable compromises to be achieved between event detection delay and lifetime without sacrificing (eventual) coverage for each point. We compare different sleep scheduling policies in terms of average detection delay, and show that ours is closest to the detection delay lower bound for stationary event surveillance. We also explain the inherent relationship between detection delay, which applies to persistent events, and detection probability, which applies to temporary events. Finally, a connectivity maintenance protocol is proposed to minimize the delay of multi-hop delivery to a base-station. The resulting sleep schedule achieves the lowest overall target surveillance delay given constraints on energy consumption.**

I. Introduction

Sensor networks promise surveillance of large areas with possibly unprecedented accuracy. Currently, energy supply is one fundamental bottleneck. It is very expensive to replace sensor node batteries once they are deployed, both because of the large number of sensing nodes and because of the typically hazardous or unfriendly environment in which these nodes are deployed. Hence, prolonging battery life is a prime consideration in network design. Current literature advocates employing redundancy to allow some nodes to go to sleep without jeopardizing sensory coverage. These approaches imply that a minimum number of nodes must remain awake for the right degree of coverage to remain satisfied. A trade-off exists between energy savings and coverage. For example, in [1], [2] partial coverage schemes are investigated to increase energy saving gains. In these efforts, both random and synchronized sleep schedules are proposed and studied for certain scenarios. The former refers to the case where each node independently chooses random sleep and wakeup times. The latter refers to the case where all nodes go to sleep and wake up together in a synchronized fashion.

In contrast, we develop a near-optimal deterministically rotating sensory coverage. In this scheme, the area is only partially covered at any point in time. However, any point is eventually sensed within a finite delay bound. The energy/coverage trade-off is thus more meaningfully expressed as one between energy savings and the average *detection delay*, defined as the average time elapsed between event occurrence at a point and its detection by a nearby sensor. It is desired to minimize average detection delay subject to a constraint on energy consumption (expressed as a duty-cycle constraint). The goal of this paper is to develop a localized distributed protocol for (near-optimally) solving the aforementioned constrained optimization problem while ensuring upper bounds on the worst-case detection delay.

The paper also addresses sleep scheduling schemes for minimizing packet delivery latency to a common base-station. Observe that at very low duty cycles, it is likely that sensor nodes that are awake at any given time do not form a connected graph unless their wakeup times are appropriately synchronized. Such synchronization, however, may deviate from the optimal sleep schedule from the perspective of minimizing average detection delay. We develop a heuristic that provides partial synchronization to reduce delivery latency without significantly impacting the average detection delay.

The combination of detection delay and packet delivery latency is the perceived surveillance delay, which refers to the time elapsed from the occurrence of an event in the system to the time the event is reported to a base-station. Hence, the overall contribution of this paper is to develop a protocol for minimizing the surveillance delay subject to energy (namely, duty cycle) constraints.

Our protocol is optimized for detection of rare (but urgent) events. In such applications, network longevity is especially important, since mission lifetime must be appropriately large. Nodes operate at very low duty cycles and do not communicate unless an event is detected. Therefore, we consider sensing power as the predominant energy drain over the system lifetime. Once detection occurs, a prompt reaction may be needed (e.g., activating a camera or reporting an emergency). Consider, for example, the detection of forest fires. There are two natural concerns with this application: first, how long will the network last once deployed? Second, how responsive will it be in reacting to fire events? Our design translates these two questions into two related design parameters; namely, the energy consumption rate (i.e., the duty cycle which determines lifetime) and the surveillance delay. Our protocol offers a design space in which the designer can trade-off these parameters in a near-optimal fashion.

The rest of the paper is organized as follows. Section II presents the general framework and assumptions underlying our approach. A localized distributed optimization algorithm is presented in Section III to produce a sleep schedule that approaches the optimal on detection delay. This algorithm is subsequently enhanced to reduce delivery latency as well. Simulation results are presented in Section IV. We survey related work in Section V, and conclude this paper in Section VI with our summary and directions for future work.

II. General Framework

We consider an area covered by sensing nodes. Let some event (e.g., a fire) occur at one point in the area. The *maximal* detection delay for an event occurring at this point is defined as the longest time that may elapse before the event is detected by a nearby node. The *average* detection delay for this point is defined as the average time elapsed until the event is detected. The maximal detection delay for the entire area is the largest value of all maximal detection delays at points in the area. Similarly, the average detection delay for the area, denoted γ, is the average value for all detection delays of all points. Trivially, when the area is sensing covered, both the maximal detection delay and the average detection delay for the area are 0, since all events are detected immediately.

Sensors in the area are duty-cycled. Most sensors have a finite "warm-up" time T_w upon startup before reliable readings can be reported. Following the warm-up time, a sensor takes a sample of the environment, which itself takes time T_s (possibly including repeated sensor readings). This may be followed by other necessary processing (such as data logging) which takes time T_p. Hence, from the instant a node is powered on, a minimum time interval, $T_{on} = T_w + T_s + T_p$, must elapse before the node can go to sleep again. Given a duty cycle constraint β which defines the maximum percentage of time a node can be awake, the node must sleep for at least a duration T_d, where $T_{on}/(T_{on} + T_d) = \beta$. Hence, any event is detected in at most $T_d + T_{on}$ time units. It is desired to minimize the *average* event detection time. We are especially interested in very low duty-cycle operation where $T_{on} << T_d$.

We propose a two-level sleep scheduling framework. The first level selects a minimal subset of all deployed nodes, called the *primary* subset, such that sensing coverage is maintained using the fewest primary nodes. We assume that there are enough nodes in the network for sensory coverage to be achieved. The remaining nodes are turned off. This process is repeated periodically at a fairly large period (e.g., of the order of tens of hours) to change the set of primary nodes so that their energy is not depleted. Algorithms for such rotation have been proposed in prior literature and are not considered in this paper. The second level focuses on the current primary nodes. It contributes further energy savings by duty-cycling these nodes at a higher frequency (e.g., seconds or minutes). That is to say, each node in the primary subset sleeps for T_d then wakes up for T_{on}, where $T_{on}/(T_{on} + T_d) = \beta$, the desired duty cycle. Our purpose, in this paper, is to coordinate the duty cycles of primary nodes such that the average detection delay in the area is minimized.

One interesting remark is that although the maximum energy savings by first level scheduling are bounded by the need to maintain sensory coverage, the second level savings can be made arbitrarily large by decreasing the duty cycle of primary nodes. In principle, there is no lower bound on energy consumption after the second level scheduling. The only consideration is that lowering the duty cycle increases average detection delay.

If the average number of primary nodes within a sensory radius is α, any point in the environment is sensed by α nodes on average. Since each node sleeps for T_d and wakes up for T_{on}, at low duty cycles (i.e., when $T_{on} << T_d$), a point is sensed on average no more than once every T_d/α time units. An event arriving randomly between sense instants will thus suffer an average detection delay no lower than $T_d/2\alpha$. This value establishes a lower bound on detection delay given the sensor wakeup period, T_{on}, and the chosen duty cycle, $\beta = T_{on}/(T_{on} + T_d)$, which uniquely determine the minimum T_d, and hence the minimum $T_d/2\alpha$.

On the other extreme, if all primary nodes sleep and wake up in unison, each point is sensed only once every T_d, and the average detection delay for a randomly arriving event is $T_d/2$. Our purpose is to design a sleep scheduling protocol that approaches the lower bound, $T_d/2\alpha$, on the average detection delay.

It can be shown that minimizing detection delay leads to minimizing the variance in detection delay as well. Intuitively, this is because the sum of the squares (or higher powers) of numbers that add up to a constant is minimized when these numbers are equal. Hence, equally spacing sensor wakeup times within an interval T_d leads to minimizing both the mean and variance of detection delay. The complete proof is omitted for space limitations.

Finally, observe a relationship between detection delay and detection probability. An event with a short lifespan can be detected as long as its lifespan intersects any of the waking periods of neighboring sensor nodes. It is easy to show that the probability of such intersection is maximized when the wakeup periods are equally spaced. Thus, the sleep scheduling that optimizes the detection delay also maximizes the detection probability of short-lived events. Next we present a protocol that produces a near-optimal sleep schedule.

III. Sleep Schedule Optimization

In this section, we describe a sleep scheduling protocol that outperforms both random and synchronized scheduling in terms of average detection delay. The protocol is distributed, and has the favorable feature that it guarantees *local optimality* in that every node ends up with a wakeup point that cannot be further improved in terms of the average detection delay within its sensing range. We also present a protocol for optimizing end-to-end delivery latency. The combination of these two protocols is explored to reduce overall surveillance delay.

A. Detection Delay Optimization

Our overall algorithm for minimizing detection delay is a three stage transition process, shown in Figure 1.

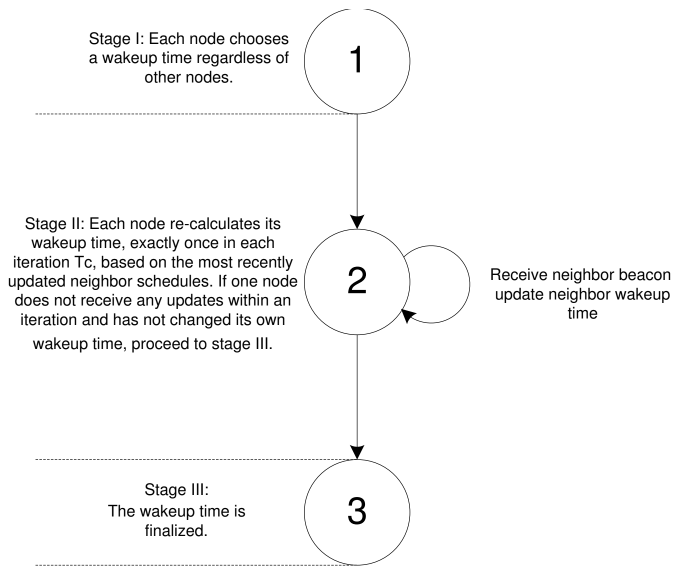
Stage I: Each node chooses a wakeup time regardless of other nodes.
1
Stage II: Each node re-calculates its wakeup time, exactly once in each iteration Tc, based on the most recently updated neighbor schedules. If one node does not receive any updates within an iteration and has not changed its own wakeup time, proceed to stage III.
2
Receive neighbor beacon update neighbor wakeup time
Stage III: The wakeup time is finalized.
3

Fig. 1. State Transition of Optimization Algorithm

We assume that neighboring nodes have approximately synchronized clocks. Protocols for clock synchronization in sensor networks can be found in [3]. Each node i starts at Stage 1, where it randomly picks an initial wakeup time, $t_i[0]$ for itself on a common timeline in the cyclic interval $[0, T_d + T_{on})$. For the purposes of this analysis, the wakeup time denotes the instant at which the node's wakeup interval T_{on} starts. The initial selection of the wakeup times of different nodes is completely uncoordinated. Each node communicates its randomly chosen wakeup time to its neighbors, sets up an iteration timer to fire at a period T_c, and enters Stage 2. Observe that in this stage all primary nodes are still awake (i.e., have not yet started their duty-cycling). The period T_c is called the *schedule iteration period*, which is different from the period $T_{on} + T_d$ of the would-be duty cycles.

In Stage 2, each node undergoes multiple schedule iterations. Within a single iteration, a node makes at most one adjustment to its wakeup time to reduce the average detection delay. Ultimately, a local minimum is reached where no more reductions can be obtained. More specifically, when the iteration timer of node i fires, denoting the beginning of a new schedule iteration, k, the node considers adjusting its wakeup time from $t_i[k-1]$ (the value chosen in the previous iteration) to a new value, $t_i[k]$. This new value should minimize the average detection delay in the area within node i's sensing range, denoted $\gamma_i[k]$, given the updated wakeup times received from i's neighbors in the last iteration. Note that by neighbors, we are only referring to those nodes that have overlapping sensing ranges with the current node, since for the the current node, only the waking times of these *sensing neighbors* are relevant. We will use *communication neighbor* to specifically refer to the nodes within communication range of the current node, and without further explanation, use neighbor to denote sensing neighbors.

In our discussion, we assume that each node knows its sensing range. This assumption is supported by our observations with current sensor nodes. For example, in XSM2 [4] motes developed by OSU and CrossBow, a roughly circular sensing range can be measured for the set of PIR sensors before deployment. Each node can use this knowledge to determine whether or not a given point is located within its sensing range.

If the difference between the old and new detection delays ($\gamma_i[k]-\gamma_i[k-1]$) is larger than a preset threshold, h, the new wakeup time, $t_i[k]$, is adopted and the node reports this new wakeup time to all its neighbors. Otherwise, the old wakeup time, $t_i[k-1]$, remains in place and no updates are sent. The node then waits for the next invocation of the iteration timer T_c to start a new iteration. If the node does not receive any updates within an iteration and has not changed its own wakeup time, it enters Stage 3 in which it starts duty-cycling, phased in accordance with its computed wakeup time. Once all nodes reach Stage 3, we consider the detection delay optimization complete. Note that, since clocks drift over time, the duty cycle period T_d+T_{on} must be large enough to accommodate a fair amount of phase drift without the need for clock re-synchronization. This constraint is met naturally, since we are interested in very low duty cycles ($T_d >> T_{on}$) in which T_d must be reasonably large (of the order of seconds or minutes).

The critical part of the above optimization process lies in the localized computation of the optimal wakeup time of an individual node at Stage 2 as a function of those of its neighbors. The problem is formulated as follows. Given a node, i, that is informed of all the current wakeup times of its neighbors, what wakeup time, $t_i[k]$, should it choose to minimize the average detection delay, $\gamma_i[k]$, in the area within its sensing range?

To answer this question, in the following, we first derive an expression for the average detection delay within the sensory range of node i as a function $f_i(t)$ of the node's unknown wakeup time t (and the known wakeup times of its neighbors). We then find the wakeup time t that minimizes this expression (i.e., for which $f_i(t)$ is minimum). Finally, we present an implementation that computes $f_i(t)$ and the corresponding wakeup time efficiently at run-time.

1) Derivation of an Optimal Wakeup Time: To derive $f_i(t)$, consider an arbitrary point A in node i's sensing range. Let point A be located within the intersection of the sensing ranges of n nodes (including node i). The average detection delay at point A is the average time elapsed from the occurrence of an event at A to the next time some neighboring node wakes up and samples the environment. It depends on the relative spacing of the respective sampling times of the n neighbors. Since each node will sample the environment once every duty cycle period, there will be a total of exactly n samples within each interval T_d+T_{on}. Let the samples of different nodes be separated by time intervals x_1, ..., x_n, where $x_j \geq 0$ for $1 \leq j \leq n$.

Figure 2 shows an example of a duty cycle of length 1, with nodes $N1$, $N2$ and O, sampling the environment at times 0.25, 0.6 and t respectively. The intervals x_1, ..., x_3 between successive samples are indicated. The circle in this figure depicts a repeated duty-cycle. The arrows indicate the direction of the passage of time. Observe that while a node might be awake for a finite period of time, T_{on} (which includes sensor warm-up and data post-processing times), its sampling time, for purposes of this analysis, refers to the time instant at which the node completes its environmental reading. In our model, this instant occurs at a fixed offset from the node's wakeup time (namely, at offset T_w+T_s defined in Section I). However, it is straightforward to extend our analysis to the case where nodes continue sampling the environment for some contiguous finite duration.

Given inter-sample separations x_1, ..., x_n, the average detection delay D at point A is given by the sum of the average detection delays for event arrivals in an interval x_j (given by, $x_j/2$), each multiplied by the probability of arriving within that respective interval, which is $x_j/(T_d+T_{on})$. Hence, D equals the sum of $(x_j/2)x_j/(T_d+T_{on})$, $1 \leq j \leq n$, which gives:

$$D = \frac{x_1^2 + ... + x_n^2}{2(T_d+T_{on})} \qquad (1)$$

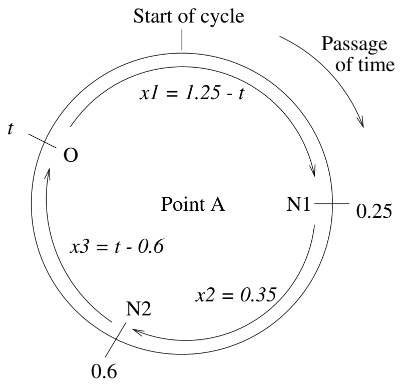

Fig. 2. A Cyclic Sleep Schedule

Since node i knows the wakeup times of all its neighbors, substituting in Equation (1) we get a quadratic expression that is a function only of node i's own wakeup time. For example, substituting with intervals x_1, x_2 and x_3, shown in Figure 2, into Equation (1) we get a quadratic function of t that represents the average detection delay at point A. Observe that this quadratic function depends on the ordering of the unknown wakeup time t with respect to the wakeup times of the neighboring nodes. For example, Figure 2 shows t to be in the range $0.6 \leq t < 1$. Substituting in Equation (1) gives an expression that is valid only for the corresponding range. Similar expressions can be derived for the other ranges. Putting the expressions for different ranges of t together, we obtain a continuous piecewise quadratic equation that yields the average detection delay at point A as a function of the unknown wakeup time t anywhere in the duty cycle. We call it the *optimality curve* for point A. The optimality curve for point A shown in Figure 2 is given in Figure 3.

To minimize the average detection delay across the entire sensing range of some node i, the quadratic optimality curves of all points in i's sensing range are added. The resulting piecewise quadratic function is the sought function $f_i(t)$ that is then solved for a global

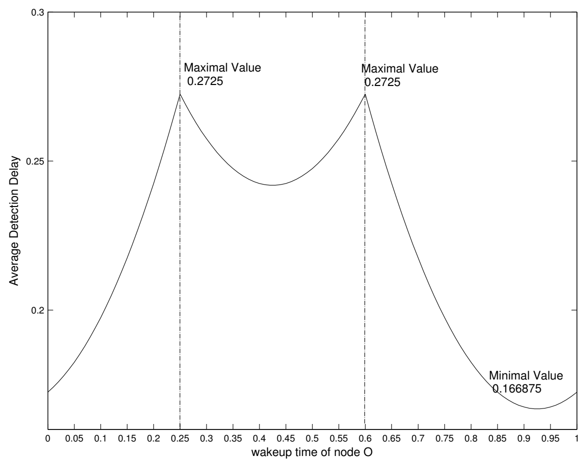

Fig. 3. Optimality Curve for node O at point A

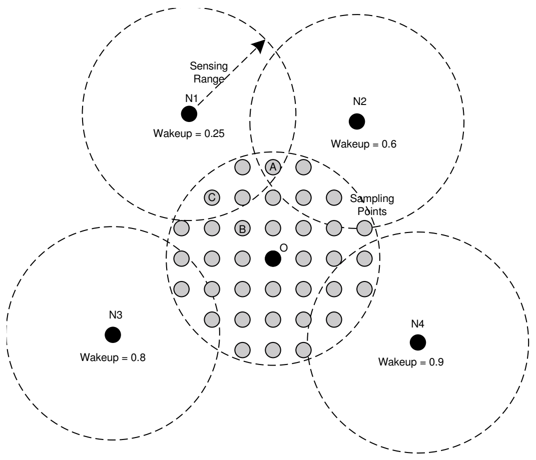

Fig. 4. An Optimization Example

minimum. This conceptual procedure lends itself to an efficient implementation in view of the following two observations.

First, note that points covered only by node i (and no other nodes) will always have the same average detection delay regardless of when i chooses to wake up. At low duty cycles, this delay is well approximated by $T_d/2$. Such points need not be considered in the aforementioned summation as they do not change the optimization result. Second, note that all points that lie at the intersection of sensing ranges of the same nodes lead to the same quadratic optimality curves. Hence, it is enough to compute such curves only once. For example, node O in Figure 4 needs to consider only five distinct optimality curves corresponding to the five intersection regions between its sensory range and that of other nodes. The equation for each curve is weighted by the area of the corresponding intersection and the results added up to obtain $f_i(t)$.

It can be shown that the resulting overall function is piecewise quadratic with a number of segments that depends only on the total number of neighbors, M, of node i. Its global minimum can only occur at one of the local minima of the individual segments or at the points at which these segments are joined. Inspecting these points is an $O(M)$ operation. The algorithm can therefore efficiently determine the position of the global minimum and hence the new wakeup time. Next, we present a detailed example of computing an optimality curve, and our actual implementation of the entire algorithm.

2) Example: Computing the Optimality Curve: Consider again node O in Figure 4. Node O has four neighbors denoted $N1$ to $N4$. In this example, there are five distinct sensor range intersection regions within O's sensing range that need to be considered. Figure 4 depicts these regions and the wakeup times of all neighboring nodes. Point A exemplifies one region that lies at the intersection of the sensing ranges of nodes $N1$, $N2$ and O. In the duty cycle $[0, 1)$, there are three cases to consider for the wakeup time t of node O, namely. $0 \le t < 0.25$, $0.25 \le t < 0.6$ and $0.6 \le t < 1$, where 0.25 and 0.6 are the known wakeup times of neighbors $N1$ and $N2$. Figure 2 depicts the case where $0.6 \le t < 1$. As seen in Figure 2, the intervals between successive wakeup times are $1.25 - t$, 0.35 and $t - 0.6$ respectively. Substituting in Equation (1), the average detection delay in this case is $\frac{(1.25-t)^2+0.35^2+(t-0.6)^2}{2}$, which evaluates to $t^2 - 1.85t + 1.0225$. Similarly, we can determine that for $0 \le t < 0.25$ the average detection delay is given by $t^2 - 1.85t + 0.4225$, and that for $0.25 \le t < 0.6$ it is given by $t^2 + 0.15t + 0.1725$. Together, the above three segments constitute the optimality curve for point A (shown in Figure 3).

3) Efficient Implementation: In our implementation, a node builds a polynomial function table for each optimality curve, in which each segment of the function is stored as a three-element tuple (a, b, c), denoting the function as $f(t) = at^2 + bt + c$, It also stores the starting and ending point of each segment. For the optimality curve computed above, the polynomial function table is shown in Table I.

A node also sorts the wakeup times of its M neighbors to determine the $M + 1$ intervals between these wakeup times within a duty cycle. It then initializes a new polynomial function table that will hold the final function $f_i(t)$ for the area covered by node i. We call it the *result* table.

To simplify computation, a node then considers a virtual grid within its sensing range. Points on this grid are considered sequentially, each with the same weight. For each point, the algorithm classifies this point based on which nodes are less than one sensing range away from it. Then, the coefficients of all segments of its optimality curve are fetched from the corresponding polynomial function table and added to the coefficients of the corresponding segments in the result table, which generates an intermediate segmented quadratic polynomial function. When all points have been considered, the result function is complete. For example, the result table for point O in Figure 4, is shown in Table II. The corresponding aggregated function is plotted in Figure 5. The optimal wakeup time can be decided by finding the lowest value on the aggregated function (which turns out to be 4.4166 at $t = 0.385$). This can be done by inspecting function values at segment boundaries and local minima (a local minimum of a function $at^2 + bt + c$ occurs at $t = -b/2a$). The time at which the lowest value occurs is the sought wakeup time $t_i[k]$ of node i at iteration k. Once the wakeup time is determined, the node sends out its decision. We now briefly explain the content of the decision packet. Each node keeps an incrementing counter as the current version of its wakeup time. It also keeps the latest versions of its neighbors. Once it makes a new adjustment, it sends out its ID, its new wakeup time, the version counter, as well as the version counters of its neighbors. The last piece of information is necessary to avoid non-serializable modifications of wakeup times of neighboring nodes. Such modifications may lead to endless loops in the adjustment. Therefore, once two nodes find that they have each adjusted their sleeping times independently, the node with lower ID

TABLE I

POLYNOMIAL FUNCTION TABLE FOR POINT A

	Range	Tuple	Function
1	$[0, 0.25]$	$(1, 0.15, 0.1725)$	$t^2 + 0.15t + 0.1725$
2	$[0.25, 0.6]$	$(1.0, -0.85, 0.4225)$	$t^2 - 0.85t + 0.4225$
3	$[0.6, 1]$	$(1.0, -1.85, 1.0225)$	$t^2 - 1.85t + 1.0225$

TABLE II

RESULT TABLE

	Range	Tuple	Function
1	$[0, 0.25]$	$(15, -4.55, 4.89)$	$15t^2 - 4.55t + 4.89$
2	$[0.25, 0.6]$	$(15, -11.55, 6.64)$	$15t^2 - 11.55t + 6.64$
3	$[0.6, 0.8]$	$(15, -18.55, 10.84)$	$15t^2 - 18.55t + 10.84$
4	$[0.8, 0.9]$	$(15, -26.55, 17.24)$	$15t^2 - 26.55t + 17.24$
5	$[0.9, 1]$	$(15, -34.55, 24.44)$	$15t^2 - 34.55t + 24.44$

revokes its prior decision and rolls back to its last version. The same rule applies to more than two nodes as well. One node also needs to roll back if its packet is lost in transmission. Therefore, we use an acknowledgement based MAC layer. If one node cannot receive the acknowledgements from all neighbors, it should either revoke its prior decision, if it receives a parallel adjustment from one of its neighbors during the time, or resend its decision to all its neighbors. Observe the fact that communication range in sensor networks is typically much larger than sensing range. Therefore, we expect that the sensing neighbors are typically located sufficiently nearby, and connected via relatively reliable links to the current node.

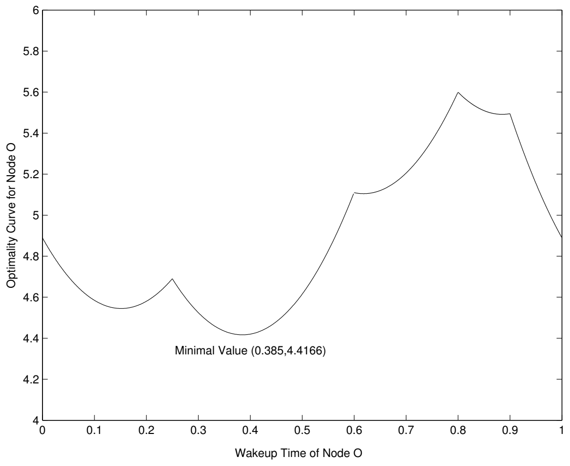

Fig. 5. Aggregated Optimality Curve for Node O

4) Algorithm Analysis: **Cost Analysis** We now consider the computational cost and requirements on storage of the algorithm. We consider storage requirements first. For each sampling point covered by n neighbors, the maximal number of segments is $n + 1$ (there is $n+1$ because we treat the first region and the last region in Figure 3 as different functions). Therefore, for a node with M neighbors, the number of segments for the aggregated function is at most $M+1$, due to the fact that points in the same partition share the same segments. Once we have aggregated a segment function, the storage it occupies can be freed, therefore, at most $M + 1$ entries are needed in the global result table, which is not memory intensive. Second, as far as computation cost goes, the overall cost is proportional to the product of grid resolution and the number of neighbors, M. We can easily control the former factor to reduce overall cost to an acceptable value. Our experiments on MICA2 nodes show that comparable computation load can be well afforded.

On the Convergence of the Algorithm We now show that the overall optimization process terminates in a finite number of steps. First, note that each adjustment of the wakeup time by one node in Stage 2 decreases the average detection delay within the sensing range of this node, but does not affect the average detection delay outside its sensing range. Hence, the average detection delay for the area also decreases with individual node adjustment. Also note that, in our design, we have avoided non-serializable adjustments of neighboring nodes. Therefore, the whole process exhibits a contractive property. Since the initial average detection delay for the whole area must be finite, and since the algorithm makes adjustments only if they decrease the average detection delay (in some node's sensing range) by some minimum finite amount, the algorithm must terminate after a finite number of adjustments. Note that during the process, it is possible that the adjustment of one node's schedule may propagate to its neighbors, however, such propagation will only decrease the overall detection delay, which obviously will terminate after a finite number of steps.

To estimate the convergence time of the algorithm in area, S, suppose each node has a sensing range, r, and communication range, $R > r$. For each adjustment of one node in Stage 2, the average detection delay decreases by at least h in the sensing area of this node. Thus, each adjustment decreases the average detection delay for the whole area by at least $\frac{\pi r^2}{S} \times h$. Remember that the average detection delay is upper-bounded by approximately $T_d/2$ and lower-bounded by approximately $T_d/2\alpha$. The maximum number of adjustments is therefore bounded by the difference between the two bounds divided by the adjustment per step, which yields $\frac{T_d}{2}(1 - 1/\alpha)\frac{S}{\pi r^2 h}$. Now assume that nodes outside each other's communication range (and hence outside each other's sensing range) can perform adjustments in parallel. There are roughly $\frac{S}{\pi R^2}$ such nodes. Hence, the number of rounds of adjustment is roughly $\frac{T_d}{2}(1 - 1/\alpha)\frac{R^2}{r^2 h}$, which takes $\frac{T_d}{2}(1 - 1/\alpha)\frac{R^2}{r^2 h}T_c$ time units to complete. This estimate, of course, is a quite relaxed bound: each adjustment may decrease the average detection delay within one node's sensing area well beyond the lower bound h. In practice, our simulations show that the system always converges within twenty rounds.

B. End-to-End Delay Optimization

Next, we propose an optimization for end-to-end delivery delay. Observe that at low duty cycles, the fraction of nodes that are awake at any given time do not necessarily form a connected network. Delivering sensed events to the base-station requires synchronization of waking times between communication neighbors along the path. We consider networks where the communication range is relatively large compared to the sensing range. Hence, after first-level scheduling (which determines the minimum number of nodes needed for full sensory coverage), the resulting primary nodes have many neighbors within their communication range. The problem, of course, is that after the ensuing second-level scheduling, not all neighbors will be awake at the same time. From the perspective of minimizing event delivery time to a base-station, it is desired to synchronize duty cycles of nodes into a streamlined sequence to pipe the data efficiently. This idea is not unlike the common practice of synchronizing traffic lights to turn green (wake up) just in time for the arrival of vehicles (packets) from previous intersections (hops). Observe that it is enough for each node to synchronize its duty cycle with only one neighbor within its communication range that is closer to the basestation. Consequently, synchronized routes are formed to expedite data delivery from any node.

An example of this type of coordination is shown in Figure 6. As shown in this example, packets delivered from node 0 to 9 have minimum delay. We call this technique *streamlined wakeup*. In the following, we propose an optimization of delivery delay based on the streamlined wakeup technique. We focus on the most common case where each sensor reports to only one base-station (although different parts of the network might report to different local base-stations). Our algorithm works as follows:

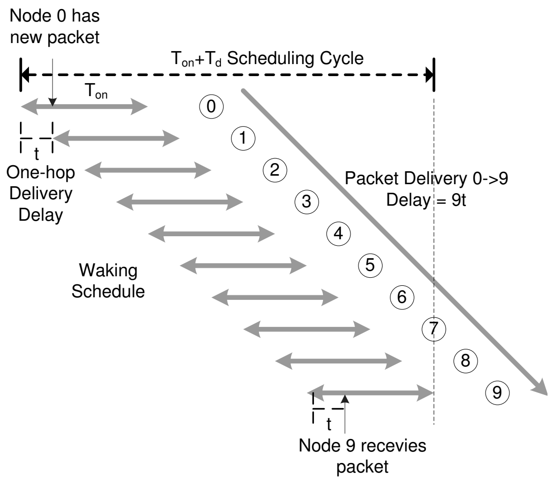

Fig. 6. Scheduling Example for Node Pipe

1) After first-level scheduling is complete, the base-station floods the network with a message containing a hop count that is incremented at each hop (interest propagation). Each node keeps track of the lowest hop-count received and maintains that number as its hop count from the base-station. Since the base-station is assumed to be always up, nodes one hop from the base-station (i.e., its direct neighbors) set a *pipe* flag indicating that they have a valid streamlined path to the destination. Any node that sets this flag communicates this fact to its neighbors.
2) Nodes run the detection delay minimization algorithm described earlier to compute their wakeup times. In Stage 3 of this algorithm, instead of actually implementing the duty cycle, they execute the step below.
3) Any node whose neighbors with shorter hop counts to the base-station have set their *pipe* flag, finds the one such neighbor with the closest wakeup time to its own. The node then overlaps its wakeup interval with that neighbor's, effectively appending itself to an established streamlined data pipe that is closest to its ideal wakeup time. Observe that the number of such pipes that may be established in the network is of the order of the number of the immediate communication neighbors of the base-station. The larger is this number, the lower (on average) is the adjustment needed to a node's wakeup time to join a pipe. For example, a base-station with a sensitive enough antenna to hear all sensors will enable each node to be its own data pipe with no additional synchronization or adjustment needed. Having joined a pipe, a node sets its *pipe* flag and communicates this fact along with its new wakeup time.
4) Any node that has set its *pipe* flag and communicated this information now enters the duty-cycling phase in according with its updated wakeup schedule.

The above algorithm ensures that a synchronization wave propagates outwards from the base-station. When the wave reaches the outer perimeter of the network, all nodes will have routes to the base-station with appropriately overlapped wakeup times. All nodes will have entered the duty-cycle mode. The initialization is thus complete. If the communication range is large enough, it is easy to find neighbors with close wakeup times to your own. The algorithm therefore does not have much impact on the optimality of average detection delay in networks with a large communication range, as will be demonstrated in the next section.

IV. Evaluation

In this section, we verify the theoretical results and optimizations given in the previous section via extensive simulations.

A. Simulation Setup

We simulate a two-level scheduling framework. By default, the area is $100m \times 100m$. Each node has a sensing range of $10m$. Initially more than enough nodes are deployed to guarantee sensing coverage. The first level scheduling is then applied where as many nodes as possible are put to sleep without compromising overall sensing coverage. The remaining nodes form the basis for evaluating the protocols designed in this paper, where different approaches are compared.

In practice, we deploy 300 nodes, followed by a first-level scheduling protocol to turn off redundant nodes. An average of 76 nodes remain awake, so we generate ten scenarios with 76 nodes remaining as the basis for second level scheduling evaluation. Each of these deployment scenarios guarantees full sensing coverage and no node is redundant. We simulated a simple MAC layer with packet acknowledgement. In the simulations, packet loss and retransmissions appear to have very limited effect on the overall performance, since we can adjust the pace of schedule readjustment sufficiently to accommodate packet retransmissions.

B. Detection Delay Optimization

In this section, we focus on the optimization of average detection delay. For each scenario, we compare the optimized and random energy saving schedules to the theoretical lower bound and the upper bound (the case of a synchronized schedule). The results are shown in Figure 7. The horizontal axis varies the ratio of the sleep interval to the waking interval, T_d/T_{on}, on a logarithmic scale, over two orders of magnitude. The vertical axis shows the normalized average detection delay over T_{on}.

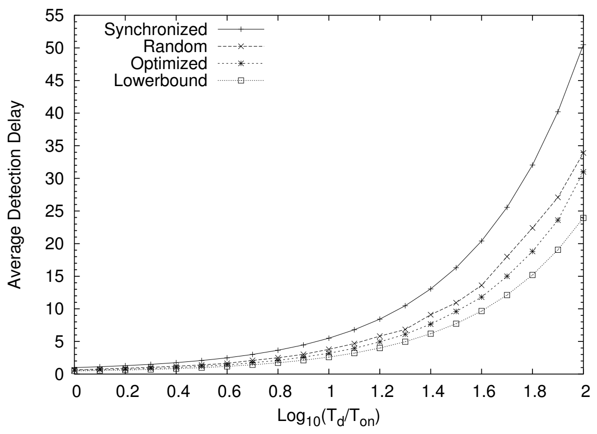

Fig. 7. Average Detection Delay

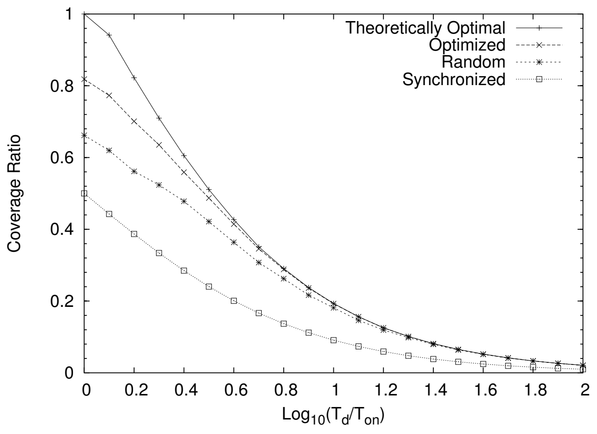

Fig. 8. Rotating Coverage Ratio

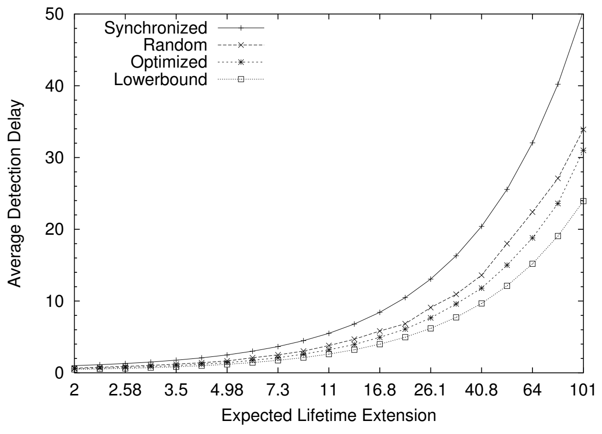

Fig. 10. The Lifetime vs. Delay Trade-off

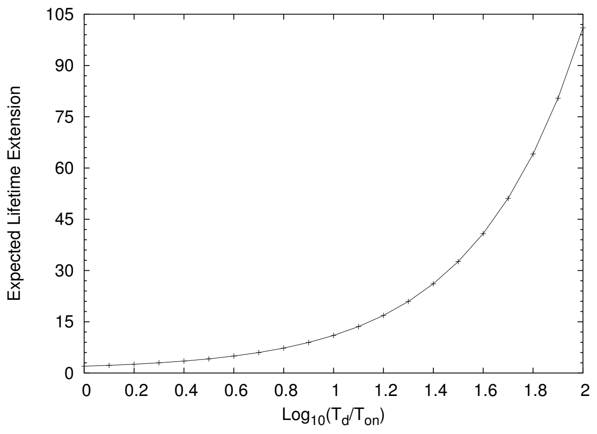

Fig. 9. Expected Lifetime Extension

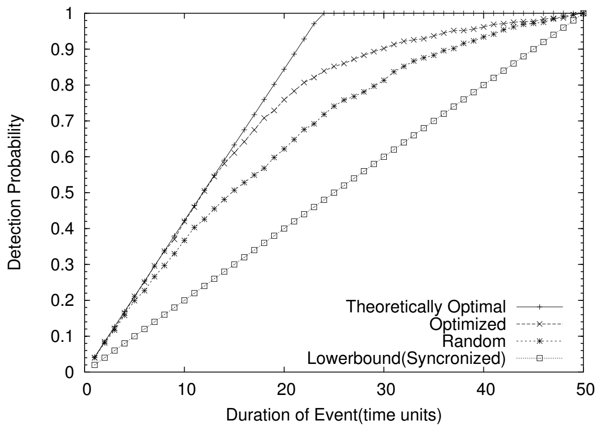

Fig. 11. Event Detection Probability

Notice that the theoretical lower bound to which we compare the results is over optimistic. No scheduling approach can achieve this bound because different nodes in an irregular network generally cannot achieve perfectly equal wakeup time spacing simultaneously. Thus, optimizing the average detection delay for one point usually leads to sub-optimal scheduling for neighboring points. While no algorithm can achieve the optimistic lower bound, we observe that ours demonstrates considerable performance enhancement compared with both random and synchronized sleep scheduling.

For example, from Figure 7, when $T_d/T_{on} = 10$ (or $log(T_d/T_{on}) = 1$), the theoretical average detection delay lower bound is 2.6, our algorithm achieves 3.2, random sleep scheduling achieves 3.8, while synchronized sleep scheduling is as high as 5.5. More generally, our protocol can reduce the gap between random scheduling and the optimal bound in terms of average detection delay by 30% to 50%, and has a absolute average detection delay reduction over random scheduling up to 15%.

We also evaluate the notion of coverage ratio defined as the percentage of covered area in time and space. For the purposes of this experiment, covered area refers to area in the range of at least one sensor that is awake at the time. Since each node is awake during T_{on}, the aggregation of such coverage intervals reflects a measure of vigilance of the network. The results are presented in Figure 8. As shown, as the duty cycle decreases (by increasing T_d/T_{on}), the coverage ratio of random scheduling and optimized scheduling converges quickly to the optimal. This is expected because both random scheduling and optimized scheduling are not likely to overlap the wakeup periods of neighboring nodes. Since the coverage ratio is only relevant to the aggregated waking period, these two sleep scheduling policies eventually lead to the same (optimal) ratio.

Another important factor is the expected extension in lifetime. Figure 9 plots the relationship between the ratio T_d/T_{on} and the expected lifetime extension of the sensor network in multiples of its original lifetime (the one when all primary nodes are always on).

Combining Figure 9 with Figure 7, we quantify the trade-off relationship between the expected lifetime extension and the corresponding increase in the average detection delay achieved by different sleep scheduling algorithms. This trade-off is expressed in Figure 10. As observed, our optimization algorithm clearly outperforms both synchronized and random scheduling in the sense of achieving a longer lifetime for the same average detection delay, or achieving a lower average detection delay for the same lifetime. This figure clearly demonstrates the advantage of our approach from an application's perspective.

At last, we present the performance of different sleep scheduling policies in detecting temporary events. If events persist for a short time duration, sleep scheduling has a profound impact on their probability of detection. Figure 11 plots the relationship between

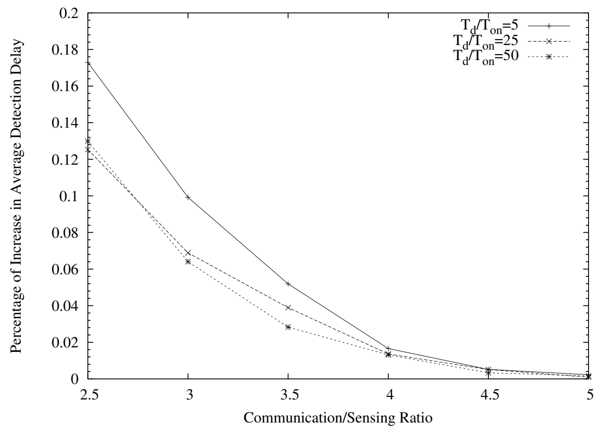

Fig. 12. Effect of Pipe Synchronization for Multihop Delivery

detection probability and event lifetime. The horizontal axis plots the event duration normalized to T_{on}, where T_{on} is assumed to be 1 time unit. In this experiment, $Td = 50T_{on}$. It is shown that our optimized sleep scheduling algorithm performs considerably better than both synchronized and random scheduling in terms of improving the probability of short event detection. This result is due to the more even spread of wakeup times under our approach.

C. *End-to-end Surveillance Delivery Latency*

Figure 12 characterizes the impact of optimizing packet delivery latency on average detection delay. The main factor that characterizes that impact is the ratio between the communication and sensing radius. Since nodes on each path to the base-station must be synchronized, their synchronization increases the average detection delay. However, as the ratio between communication range and sensing ranges increases, the number of primary nodes within one's communication range increases, which makes it easier to find a neighbor to synchronize with. The negative effect of such synchronization on average detection delay is thus reduced.

At last, we want to emphasize that while our algorithm is locally optimal, it has left a gap between itself and the theoretical global optimal. More global coordination of sleep schedules may improve performance further. We believe, however, that it would be difficult to beat this performance with other *localized* algorithms.

V. Related Work

Minimizing energy consumption has been a central topic in many papers in recent years. Effective techniques have been proposed to put nodes to sleep while maintaining full coverage at a specified degree of redundancy [5], [6], [7], [8]. These solutions can be conveniently integrated as first level scheduling algorithms in our framework.

Research on partial coverage based protocols has received less attention. Among the first publications are [2] and [1], which study the problem of tracking moving targets. Our work differs in that (i) we focus on stationary event detection, and (ii) we aim at finding a localized algorithm that approaches the minimum average delay bound.

In studying the impact of partial sensing coverage, we inevitably face the problem of connectivity. The work of [9] proposes remote radio triggered hardware, which extracts energy from specific radio signals without using an internal energy source, to provide wakeup signals. This service can be used in our framework to forcefully wake up additional nodes to form a connected network when an event of interest is detected. Since we focus on rare (but important) events, wasting energy when an event occurs is permissible. A similar hardware is reported in [10], where a low-power VLSI wake-up detector is designed in an acoustic surveillance sensor network.

Also relevant to our analysis for delivery latency is [11], which addresses this issue through an extension of first passage percolation theory for completely uncoordinated scheduling. Similarly, [12] addresses this issue through a Markov model based approach, where distribution of the data delivery delay is analytically determined. [13] uses a slotted approach for communication scheduling, where nodes determine their wakeup times based on their relative positions in the aggregation tree. Our work is different from these efforts primarily in our streamlined wakeup scheduling, as discussed in Section III-B, where the delivery latency is considerably lower.

VI. Conclusion and Future Work

In this paper, we have outlined, studied and evaluated the problem of minimizing surveillance delay subject to energy constraints. We consider this delay to be composed of detection delay and delivery delay, and propose optimizations for both. The final outcome is a flexible framework in which application designers can trade-off energy versus latency of event detection. We focus on detection of rare events, where the network is normally silent, except when events occur. This is in contrast to data collection networks that continuously stream periodic data to a collection center. The study reported in this paper is a first step towards more general models that optimize performance in the presence of communication as well. A general study of optimizing detection delay for moving targets is another worthy extension. We expect to address these issues in future publications.

Acknowledgements

The work reported in this paper was supported in part by the National Science Foundation under grants EHS-0208769 and ITR EIA-0205327.

References

[1] S. Ren, Q. Li, H.N.Wang, X. Chen, and X. Zhang, "Probabilistic coverage for object tracking in sensor networks," in *Mobicom 2004 Poster Session*, 2004.
[2] C. Gui and P. Mohapatra, "Power conservation and quality of surveillance in target tracking sensor networks," in *ACM Mobicom*, 2004.
[3] M.Maroti, B.Kusy, G.Simon, and A.Ledeczi, "The flooding time synchronization protocol," in *ACM Sensys*, 2004.
[4] "Xsm website http://cast.cse.ohio-state.edu/exscal/."
[5] X. W. et al., "Integrated coverage and connectivity configuration in wireless sensor networks," in *ACM SenSys*, 2003.
[6] T. Yan, T. He, and J. A. Stankovic, "Differentiated surveillance for sensor networks," in *ACM SenSys*, 2003.
[7] D. Tian and N.D.Georganas, "A node scheduling scheme for energy conservation in large wireless sensor networks," in *Wireless Communications and Mobile Computing Journal*, 2003.
[8] T.He, S.Krishnamurthy, J.A.Stankovic, T. Abdelzaher, L. Luo, R. Stoleru, T. Yan, L. Gu, G. Zhou, J. Hui, and B. Krogh, "Vigilnet:an integrated sensor network system for energy-efficient surveillance," in *In submission to ACM Transaction on Sensor Networks*, 2004.
[9] L. Gu and J. Stankovic, "Radio-triggered wake-up capability for sensor networks," in *IEEE RTAS*, 2004.
[10] D. Goldberg, A.Andreou, P.Julian, P.Pouliquen, L.Riddle, and R.Rosasco, "A wake-up detector for an acoustic surveillance sensor network: Algorithm and vlsi implementation," in *IEEE IPSN*, 2004.
[11] O. Dousse, P. Mannersalo, and P. Thiran, "Latency of wireless sensor networks with uncoordinated power saving mechanisms," in *MobiHoc*, 2004.
[12] C.F.Chiasserini and M.Garetto, "Modeling the performance of wireless sensor networks," in *IEEE Infocom*, 2004.
[13] "Tinydb project http://telegraph.cs.berkeley.edu/tinydb/."