Delay Efficient Sleep Scheduling in Wireless Sensor Networks

Gang Lu*, Narayanan Sadagopan†, Bhaskar Krishnamachari*†, Ashish Goel‡
*Department of Electrical Engineering
†Department of Computer Science
University of Southern California, Los Angeles, 90089
‡Department of Management Science and Engineering
Stanford University, Stanford CA 94305-4026
{ganglu, narayans, bkrishna}@usc.edu,ashishg@stanford.edu

***Abstract*— Medium access techniques for wireless sensor networks raise the important question of providing periodic energy-efficient radio sleep cycles while minimizing the end-to-end communication delays. This study aims to minimize the communication latency given that each sensor has a duty cycling requirement of being awake for only $\frac{1}{k}$ time slots on an average. As a first step we consider the single wake-up schedule case, where each sensor can choose exactly one of the k slots to wake up. We formulate a novel graph-theoretical abstraction of this problem in the general setting of a low-traffic wireless sensor network with arbitrary communication flows and prove that minimizing the end-to-end communication delays is in general NP-hard. However, we are able to derive and analyze optimal solutions for two special cases: tree topologies and ring topologies. Several heuristics for arbitrary topologies are proposed and evaluated by simulations. Our simulations suggest that distributed heuristics may perform poorly because of the global nature of the constraints involved. We also show that by carefully choosing multiple wake-up slots for each sensor significant delay savings can be obtained over the single wake-up schedule case while maintaining the same duty cycling. Using this technique, we propose algorithms that offer a desirable bound of $d+O(k)$ on the delay for specialized topologies like the tree and grid and a weaker guarantee of $O((d+k)\log n)$ for arbitrary graphs, where d is the shortest path between 2 nodes in the underlying topology and n is the total number of nodes.**

Keywords: Mathematical Optimization, Graph Theory, Combinatorics, System Design

I. Introduction

Wireless sensor networks (WSN) are expected to operate for months if not years on small inexpensive batteries with limited lifetimes. Therefore energy efficiency is typically the primary goal in these networks. Previous works have identified idle listening of the radio as a major source of energy wastage (e.g. [1], [3], [4], [5], [6], [7], [8]). Measurements on existing sensor device radios show that idle listening consumes nearly the same power as receiving. In sensor network applications where the traffic load is very light most of the time, it is therefore desirable to turn off the radio when a node does not participate in any data delivery.

The S-MAC medium access protocol (presented in [1], [2]) introduced synchronized periodic duty cycling of sensor nodes as a mechanism to reduce the idle listening energy cost. In S-MAC each node follows a periodic active/sleep schedule, synchronized with its neighboring nodes. During sleep periods, the radios are completely turned off, and during active periods, they are turned back on to transmit and receive messages. Although the synchronized low duty cycle operation of a sensor network is energy efficient, it has one major deficiency: it increases the packet delivery latency. At a source node, a sampling reading may occur during the sleep period and has to be queued until the active period. An intermediate node may have to wait until the receiver wakes up before it can forward a packet received from its previous hop. This is called *sleep latency* ([1]), and if all nodes are synchronized to the same schedule, it increases proportionally with hop length by a slope of schedule length (active period plus sleep period).

In scenarios where minimizing sleep latency is not important (non time critical applications), [9] also present an excellent analysis on bounds on the delay of sending data from a node to a sink using a completely decentralized duty cycling scheme. They show that if each sensor turns on and off independent of the other sensors, the delay incurred is proportional to the distance of the node from the sink. However the rate of this linear increase is not dependent on the locations of the nodes, but on the node density, transmission range and the average active and sleep durations.

The question arises whether energy-efficient duty cycling may be maintained while reducing sleep latency. One approach to this is the use of *adaptive listening* where nodes that lie one or more steps ahead in the path of a transmission can be kept awake for an additional length of time (present as an extension to the basic S-MAC in [2], as well as the T-MAC protocol [3]). This approach provides some reduction in sleep latency at the expense of greater energy expense due to extended activation and overhearing, but is not sufficient for long paths.

In a recent work [10], we investigated an alternate approach to delay-efficient sleep scheduling, designed specifically for wireless sensor networks where the communication pattern is restricted to an established unidirectional data gathering tree. In this case, we showed that the sleep latency can be essentially eliminated by having a periodic receive-transmit-sleep cycle with level-by-level offset schedules, in which data cascades in step by step from the leaves of the tree towards the sink, with nodes going to sleep as soon as they transmit their packets to the next level, and waking up just in time to receive the next round of packets.

In this paper we seek to address a more general and harder version of this problem: *how should the activity of sensor radio nodes be scheduled in arbitrary network communication topologies, in order to minimize the sleep latency while providing energy efficiency through periodic sleep?* This is clearly an issue of fundamental significance in the area of wireless sensor networks, and to our knowledge has never been investigated before. Unlike prior work in this area, which has focused primarily on designing new sensor network MAC protocols in an intuitive manner, we shall take an algorithmic approach.

The rest of the paper is organized as follows: We first discuss the problem scenario and the assumptions made in this study (in section II). We define a graph-theoretic combinatorial optimization problem formulation for delay efficient sleep scheduling (in section III) for the single wake up schedule case where each sensor chooses exactly one of the k slots to wake up. We show (in section IV) that this problem is in fact NP-hard in general. However, we are able to derive and analyze optimal solutions for some special cases, namely a ring topology and any tree topology. For arbitrary topologies, we propose several heuristics in section V and evaluate them using simulations in section VI. In section VII, we show that a careful choice of multiple wake up slots for each sensor offers significant delay savings over the single schedule case with the same duty-cycling of $\frac{1}{k}$. Using this technique we propose algorithms with provable delay guarantees for grid, tree and arbitrary topologies. Finally, we shall conclude with a summary and discussion of this work as well as future extensions (in section VIII).

II. Problem Scenario and Assumptions

In sensor networks with light traffic load, duty cycling (where sensors turn off their radios when not needed) is a very useful technique for reducing the energy consumption due to idle listening. We use k as a parameter that captures the duty cycling requirements of an application. To achieve the requisite duty cycling, a sensor should be kept awake on an average for $\frac{1}{k}$ fraction of the time slots. We initially focus on the single wake up schedule case, where the schedule length is k slots and each sensor is assigned one of the k slots during which it activates its radio for reception (known as the active slot), while it can potentially transmit at any slot if it has a packet to be forwarded. If a node has to forward a packet to its neighbor, it can wake up at the active reception slot of that neighbor and transmit the packet. This conserves energy of both the transmitting and the receiving node. Figure 1 shows a couple of slot assignments on a network and the resulting delays on each link. Consider figure 1 (b). Assume that node A has a packet to send to node F. A would have received this packet in slot 0, but can only transmit to E at slot 1. Thus the delay from A to E is 1 (as A waits for the complete reception of the packet at slot 0). Similarly E can only forward the packet to F in slot 2, thus incurring a delay of 1 from E to F. In this case the end to end delivery latency is 2. Ideally, if every pair of nodes can have a path on which

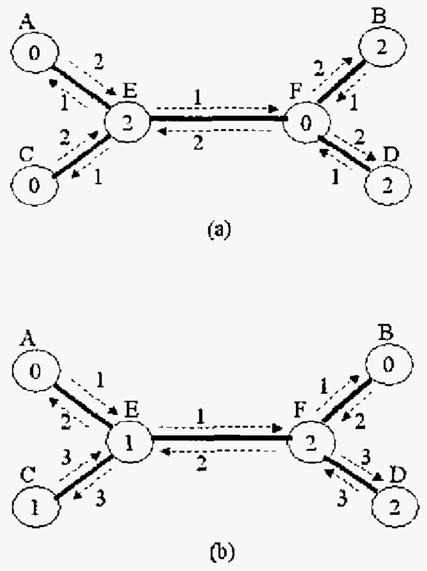

Fig. 1. Examples of slot assignment with $k = 3$. The dotted arrows show the delay on each link in the corresponding direction.

all nodes have sequentially increasing slots (modulo k), the latency will only be the number of hops between them times a single slot length ($\frac{1}{k}$-th of the schedule length). However a scheme such as the basic S-MAC scheme which synchronizes all nodes to have the same cycle will have a latency as large as the number of hops times the duration of a full period. As mentioned in section I, DMAC can achieve the ideal case for any source to sink communication path for a unidirectional data gathering tree. However, this study addresses the issue of assigning slots to minimize the maximum delay between nodes that can communicate in an arbitrary pattern. Clearly as seen in figure 1, different slot assignments to the nodes in the network could result in significantly different path delays.

Before formally defining the problem, we describe our assumptions:

- **Synchronization:** None of the discussion about sleep scheduling would be relevant if there were not some mechanism to provide time synchronization in the sensor network. However, techniques capable of providing micro-second level synchronization have been developed for sensor networks [13], [14], [15].
- **Low Traffic:** We have assumed that there is very low traffic within the sensor network. This is reasonable in low-data-rate sensor networks where phenomena of interest occur rarely. Energy-efficient low-duty cycles are only possible if this assumption is true. It also justifies the fact that this problem formulation does not take into account any queueing latency due to congestion, or significant interference/collisions (though random access schemes may be implemented to handle occasional contention during the active periods, as in S-MAC). Since interference is not a primary concern in light traffic, we have not incorporated any local vertex/edge coloring

constraints into our problem formulation which would be necessary for graph-coloring based TDMA scheduled access mechanisms such as [16].

- **Packet-Length Slot:** A related assumption we make is that for such a low traffic scenario each reception slot is of a fixed length that is sufficient only for the transfer of a single packet. Thus a packet may travel at most one hop in a single slot. Longer fixed slot lengths would not be energy-efficient if traffic is low.
- **Graph Abstraction:** While several recent papers in sensor networks (e.g. [17], [19], [18], [20]) have shown that wireless links can be quite unreliable and vary significantly in packet reception rates in each direction, we have used a binary-link-based graph-theoretic problem formulation in this work. This is justified because the communication graph we are referring to is not necessarily the full wireless network, but a logical topology which can be constructed, for instance, by filtering or blacklisting out all unreliable/unidirectional links. Others have suggested that such blacklisting is necessary for reliable packet delivery in any case [19].
- **Arbitrary Communication Pattern:** In sensor networks where the traffic is restricted to data gathering from all nodes to a single sink, it is not necessary to minimize the delay diameter between any two nodes in the graph. However, this unidirectional traffic pattern is a special case which has been addressed previously in the DMAC work [10]. In more sophisticated embedded wireless sensor networks, which may involve complex patterns of in-network processing, or communication between sensors as well as actuators, other traffic patterns are possible. We formulate the problem for the more general case in section III-A, which as we shall show is in fact computationally harder. Although we do not treat it in depth here, an alternative formulation that can provide some way of weighing different application-specific traffic patterns is also defined in section III-B.
- **Fixed Number of Slots:** In our formulation, we assume that the number of slots available to the network is fixed. This essentially defines the duration of the periodic sleep cycle, and the duty-cycle, which are assumed to be determined *a priori* by application-specific needs for energy efficiency as well as limitations on sleep/wakeup times of the radio hardware involved. As we shall see, generally with a larger number of available slots, the energy efficiency is higher but the end-to-end delay is also longer.
- **Energy Conservation:** Sensor node radios incur differing energy costs in idle listening, receiving and transmission modes. Transmission costs are generally higher than idle/reception costs. Technically, the minimum delay path obtained may involve longer hops (more transmissions) than the minimum hop-count path on the original graph. Thus delay minimization can result in a slight increase in the energy costs, however we believe this is a second order effect since the bulk of the energy savings in the network are provided by the sleep mode of the radio.

In section, III, we formally define the problem of assigning slots to nodes to minimize the network delay.

III. Problem Definition

Let $G = (V, E)$ be an arbitrary graph. Let k be the parameter that dictates the duty cycling requirements. As mentioned in section II, we initially focus on the single wake up schedule case where the schedule length is k slots and each sensor is assigned one of these k slots. Assigning a slot $s \in [0 \cdots k-1]$ to a node i schedules i to wake up (activate its radio for receiving) only at slot s. While i can transmit at any slot, it can only receive data at the beginning of slot s. Let $f : V \rightarrow [0 \cdots k-1]$ be a slot assignment function that assigns a slot to every node in the graph. Clearly f determines the delay incurred in transmitting data from one node to the other. For a given f, let $d_f(i,j)$ be the delay in transmitting data from i to j where $(i,j) \in E$:

$$d_f(i,j) \;=\; \begin{cases} k \quad (\text{if } f(i) = f(j)) \\ (f(j) - f(i)) \mod k \quad (\text{otherwise}) \end{cases} \tag{1}$$

From the definition above, it also follows that:

$$d_f(i,j) + d_f(j,i) \;=\; \begin{cases} k \quad (\text{if } f(i) \neq f(j)) \\ 2k \quad (\text{otherwise}) \end{cases} \tag{2}$$

Delay on a path P under a slot assignment f is defined as

$$d_f(P) \;=\; \sum_{(i,j)\in P} d_f(i,j) \tag{3}$$

As seen from the above discussion, duty cycling requirements will lead to increased delays in the network. We consider the following scenarios:

A. *All to All Communication*

In this scenario, every pair of sensors is equally likely to communicate. Hence, it is desirable to assign slots to the nodes such that no two nodes incur arbitrarily long delays in communication. We characterize this network wide delay using the following definition:

Definition 1: **Delay diameter** (D_f): For a given graph $G = (V, E)$, number of slots k and a slot assignment function $f : V \rightarrow [0 \cdots k-1]$, the *delay diameter* is defined as $\max_{i,j \in V} P_f(i,j)$, where $P_f(i,j)$ is the delay along the shortest delay path between nodes i and j under the given slot assignment function f.

In figure 1(a), the *delay diameter* is 5, while in (b) it is 8 (path D-F-E-C). Thus, in *all to all* communication, our design goal is given as follows:

Definition 2: **Delay Efficient Sleep Scheduling (DESS):** Given a graph $G = (V, E)$ and the number of slots k, find an assignment function $f : V \rightarrow [0 \cdots k-1]$ that minimizes the *delay diameter* i.e.

$$f \;=\; \arg\min_{f'}\{D_{f'}\} \tag{4}$$

B. *Weighted Communication*

In this scenario, the frequency of communication between a pair of sensors is not the same across all pairs. This may happen in the case of a hierarchical network structure (like clustering). Here, it would be of interest to minimize the average delay in the network, which is defined as follows:

***Definition** 3:* **Average Delay diameter** (D_f^{avg}): For a given graph $G = (V, E)$, number of slots k, a slot assignment function $f : V \rightarrow [0 \cdots k-1]$ and weights $w(i,j) \geq 0$, the *average delay diameter* is defined as $\sum_{i,j \in V} w_{ij} * P_f(i,j)$, where $P_f(i,j)$ is the delay along the shortest delay path between nodes i and j under the given slot assignment function f.

In *weighted communication*, our design goal is the following:

***Definition** 4:* **Average Delay Efficient Sleep Scheduling (ADESS)** Given a graph $G = (V, E)$, the number of slots k, weights $w(i,j) \geq 0$, find an assignment function $f : V \rightarrow [0, \cdots k-1]$ that minimizes the *average delay diameter* i.e.

$$f = \arg\min_{f'}\{D_{f'}^{avg}\} \tag{5}$$

Intuitively, in both DESS and ADESS, the objective is to color a graph with the given k colors such that the desired global objective (minimizing the *delay diameter* in the former and the *average delay diameter* in the latter) is achieved. The reader may perceive a connection to the well-known NP-complete graph coloring problem [11], which deals with minimizing the number of colors needed to ensure that no two adjacent vertices are colored the same. However, a key difference between the graph coloring problem and DESS (or ADESS) is that the former is essentially about a local constraint (adjacent vertices requiring distinct colors), while the latter is inherently more global in nature: adjacent vertices may share the same slot assignment but the maximum of the shortest delay paths between *all pairs* of nodes must be reduced. We will show below that both DESS and ADESS are also NP-Complete.

IV. Analysis

We first prove that the decision problem corresponding to DESS is NP-Complete by reduction from 3- Conjunctive Normal Form - Satisfiability (3-CNF-SAT). We also show how this reduction can be used to show that the decision version of ADESS is also NP-Complete. For two specific topologies (tree and ring), we formally characterize the optimal solution for DESS. We then show how the optimal solution for a ring may form a basic building block for an optimal assignment for cyclic graphs using the grid topology as an example.

A. *NP-Completeness*

We first define the decision problems for both DESS and ADESS.

***Definition** 5:* $DESS(G, k, f, \Delta)$: Given a graph $G = (V, E)$, number of slots k, a positive number Δ and a slot assignment function $f : V \rightarrow [0, \cdots k-1]$, is $D_f \leq \Delta$.

***Definition** 6:* $ADESS(G, k, f, w, \Delta)$: Given a graph $G = (V, E)$, number of slots k, a positive number Δ, a slot assignment function $f : V \rightarrow [0, \cdots k-1]$, and positive weights w_{ij} for all $i, j \in V$, is $D_f^{avg} \leq \Delta$.

We now prove the main complexity result:

***Theorem** 1:* $DESS(G, k, f, \Delta)$ is NP-Complete.

Proof: Given the slot assignment function f, one can compute the shortest delay path from each node to all the other nodes in polynomial time. Moreover, there are only a polynomial number of such nodes. The maximum delay among all the pairwise paths should then be compared against Δ. All these steps can be done in polynomial time. Thus $DESS(G, k, f, \Delta) \in NP$.

To prove that $DESS(G, k, f, \Delta)$ is NP-Hard, we show a polynomial time reduction from 3-CNF-SAT to DESS(G,2,f',4) (which is a special case of $DESS(G, k, f, \Delta)$ with $k = 2$, $\Delta = 4$ and f' which is defined later). This reduction is similar to the one used for showing that constructing a minimum energy broadcasting tree in a wireless ad hoc network is NP-Hard [12].

Consider a 3-CNF formula F consisting of n clauses and m literals i.e. $F = c_1 \bigwedge c_2 \bigwedge \cdots c_n$, where each $c_i = x_{i1} \bigvee x_{i2} \bigvee x_{i3}$ and $x_{ij} \in \{x_1, \bar{x_1}, \cdots x_m, \bar{x_m}\}$. For non-triviality, we assume that a clause does not contain a literal and its complement (as such a clause is trivially satisfiable). Given a 3-CNF formula F, construct a graph $G = (V, E)$ are follows:

1) $S \in V$.
2) For each variable x_i: X_i, X_{i1} (representing x_i), and X_{i2} (representing $\bar{x}_i$) $\in V$.
3) For each clause c_i: $C_i \in V$.
4) $\forall i \in [1, \cdots m]$: (S, X_{i1}) and $(S, X_{i2}) \in E$.
5) $\forall i \in [1, \cdots m]$: (X_i, X_{i1}) and $(X_i, X_{i2}) \in E$.
6) If x_i appears in c_j, $(X_{i1}, C_j) \in E$. If $\bar{x}_i$ appears in c_j, $(X_{i2}, C_j) \in E$.

The diameter (number of hops) of G is 4. Consider the following slot assignment function f':

1) $f'(S) = 1$ i.e. S wakes up only at slot 1.
2) $\forall i \in [1, \cdots m] : f'(X_i) = 1$.
3) $\forall j \in [1, \cdots n] : f'(C_j) = 1$.
4) $f'(X_{i1}) = 0$ iff x_i is true, else $f'(X_{i1}) = 1$. Moreover, $f'(X_{i1}) + f'(X_{i2}) = 1$.

Since $k = 2$, $d_{f'}(i,j) = d_{f'}(j,i) = 1$ iff $f'(i) \neq f'(j)$. If $f'(i) = f'(j)$, then $d_{f'}(i,j) = d_{f'}(j,i) = 2$

This reduction can be computed in polynomial time. Figure 2 illustrates the reduction for a given formula F. We will now show that a formula F is satisfiable iff $D_{f'} \leq 4$ in G.

If the formula F is satisfiable, for every clause c_i, at least one literal x_j is true. Thus for every node C_i in G, there exists a node X_{jk} ($k = 1$ or $k = 2$) such that $f'(X_{jk}) = 0$. Thus, we can make the following observations about the delays along the paths from various nodes to S:

1) $\forall i \in [1, \cdots n] : d_{f'}(C_i \rightarrow S) = d_{f'}(S \rightarrow C_i) \leq 2$ i.e. there exists a path from $C_i \rightarrow S$ and from $S \rightarrow C_i$ that incurs a delay of 2. Such a path is $C_i \rightarrow X_{jk} \rightarrow S$ (and vice versa) which has an alternating $0-1$ slot assignment.
2) $\forall j \in [1, \cdots m] : d_{f'}(X_j \rightarrow S) = d_{f'}(S \rightarrow X_j) \leq 2$. i.e. it is possible to reach S from every X_j by incurring a

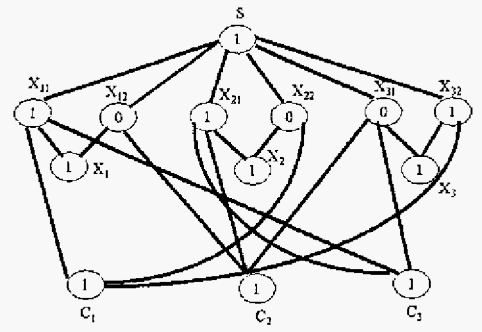

Fig. 2. Reduction from 3-CNF-SAT to $DESS(G, 2, f', 4)$. Here, $F = (x_1 \bigvee \bar{x_2} \bigvee \bar{x_3}) \bigwedge (\bar{x_1} \bigvee x_2 \bigvee x_3) \bigwedge (x_1 \bigvee x_2 \bigvee x_3)$. The satisfying assignment is $x_1 = 0$, $x_2 = 0$ and $x_3 = 1$.

delay of 2. This is because for each X_j, there exists an X_{jk} such that $f'(X_{jk} = 0)$ (since $f'(X_{j1}) + f'(X_{j2}) = 1$). A possible path from X_j to S is $X_j \rightarrow X_{jk} \rightarrow S$.

3) $\forall j$ ● $[1, \cdots m]$: $\max\{d_{f'}(X_{j1}, S), d_{f'}(X_{j2}, S)\} = 2$. This is because exactly one of $f'(X_{j1})$ or $f'(X_{j2})$ will be 1 and $f'(S) = 1$.

Thus, for any given pair of nodes a and b, the maximum delay incurred on a path from $a \rightarrow S \rightarrow b$ is at most 4. Hence, $D_{f'} \leq 4$ (which is also the hop diameter of G).

If the formula F is not satisfiable, there exists at least one clause c_i such that none of its literals are true. Thus, $d_{f'}(C_i, X_{jk}) = d_{f'}(X_{jk}, C_i) = 2$, for all $(C_i, X_{jk}) \in E$ (since $f'(C_i) = f'(X_{jk}) = 1$). Now, let y_l be a literal that appears in c_i. Consider a path from C_i to the node X_{lp} (where X_{lp} is the node that represents the complement of y_l. For example, if $y_l = x_z$, then $p = 2$. If $y_l = \bar{x_z}$, then $p = 1$). Every path from C_i will reach a vertex X_{jk} (such that the corresponding variable x_{jk} appears in c_i) for which $f'(X_{jk}) = 1$. This first hop will incur a delay of 2. From X_{jk}, one can either go to S ($f'(S) = 1$) or a C_j ($f'(C_j) = 1$) or X_j ($f'(X_j) = 1$). This hop also incurs a delay of 2. At least one more edge has to be traversed to reach node X_{lp}, which has a delay of at least 1. Thus, there exists 2 nodes C_i and X_{lp} such that the shortest delay path between them has a delay of at least 5. Thus, $D_{f'} > 4$.

Hence $DESS(G, k, f, \Delta)$ is NP-Hard. ■

As a corollary, we state the following:

Corollary 1: $ADESS(G, k, f, w, \Delta)$ is NP-Complete.

Proof: If the weights w_{ij} are uniform for all pairs i and j, then the same reduction as above can be used to show the NP-Completeness of $ADESS(G, k, f, w, \Delta)$. ■

For the rest of the paper, we will focus only on DESS. Although it seems unlikely that efficient algorithms will exist for DESS (or ADESS) on arbitrary graphs, in section IV-B we show how DESS can be efficiently solved for two specific topologies (tree and ring).

B. *Optimal Assignment on Specific Topologies*

In this section, we formally characterize the optimal assignment function f (that minimizes the *delay diameter* D_f) for 2 specific topologies: tree and ring. Using results from simulated annealing on a grid, we also show how an optimal assignment for a ring might form a basic building block of a good assignment on cyclic graphs.

1) Optimal Assignment on a Tree:

Theorem 2: Consider a tree $T = (V, E)$. Let the number of slots be k. Let the diameter of T (in hops) be h (from node a to b, say). Then for every slot assignment $f : V \rightarrow [0, \cdots k-1]$, $D_f \geq \frac{hk}{2}$.

Proof: Consider a path between two nodes p to q having x hops. Since T is a tree, this is the only path between p and q. Consider an arbitrary slot assignment function $f : V \rightarrow [0, \cdots k-1]$. Now,

$$\begin{aligned} d_f(p \rightarrow q) &= \sum_{j=1}^{x} d_f(i_j, i_{j+1}) \\ d_f(q \rightarrow p) &= \sum_{j=1}^{x} (k - d_f(i_j, i_{j+1})) \end{aligned}$$

Thus,

$$\begin{aligned} d_f(p \rightarrow q) + d_f(q \rightarrow p) &= kx. \\ \max\{d_f(p \rightarrow q), d_f(q \rightarrow p)\} &\geq \frac{kx}{2} \end{aligned} \quad (6)$$

This is true for each pair of nodes including a and b. Thus, for every slot assignment function f, $D_f \geq \frac{hk}{2}$, where h is the diameter of T. ■

Based on theorem 2, the following assignment function f will minimize the *delay diameter* of the tree $T = (V, E)$ whose hop diameter is h (from a to b): Just use 2 slot values, 0 and $\lceil \frac{k}{2} \rceil$. Let $d_f(a) = 0$. Adjacent vertices are assigned different slots (similar to a chess board pattern). In this case $\forall i, j : (i, j) \in E : \max\{d_f(i, j) = d_f(j, i)\} = \lceil \frac{k}{2} \rceil$. Hence $\max\{d_f(a \rightarrow b), d_f(b \rightarrow a)\} = \lceil \frac{hk}{2} \rceil$, which tightly matches the lower bound on the *delay diameter* of T. Thus, an optimal slot assignment for a tree balances the delay in each direction along a path as shown in figure 1(a).

2) Optimal Assignment on a Ring: We first show the optimal assignment for the case where the number of nodes n on a ring is a multiple of the number of slots k i.e. $n = mk$. We then present a lower bound for the case when the number of nodes is not an exact multiple.

Theorem 3: Consider $n = mk$ nodes $0, 1, \cdots mk - 1$ arranged on a ring in the clockwise direction. The optimal slot assignment function f is specified as follows: $f(0) = 0$. $\forall i : 1 \leq i \leq mk - 1 : f(i) = (f(i-1) + 1) \mod k$.

Proof: We will refer to such an f as the sequential slot assignment as it assigns a sequentially increasing slot (modulo k) to the nodes around the ring (see figure 6 (a)). We prove theorem 3 by contradiction. For $k = 2$, it is easy to show that assigning 2 adjacent nodes the same slot incurs a delay of 2 in both directions on that link, while a sequential assignment will yield a delay of 1 in either direction. Hence, we focus on the case where $k \geq 3$. For a sequential slot assignment f, it is easy to show that the *delay diameter* is given by:

$$D_f = m(k-1) \quad (7)$$

Assume that there exists a slot assignment function f', such that $D_{f'} < D_f$. In the rest of the proof, we will focus on the delay in the ring due to f'.

Consider a block of m links on the ring from node 0 to node m as shown in figure 3. Since we assumed that $D_{f'} < m(k-1)$, the shortest delay path from node 0 to node m (and vice versa) must lie completely within the block. The alternative path has $m(k-1)$ links each incurring a delay of at least 1 (If this alternative path is the shortest delay path, it contradicts our assumption that $D_{f'} < m(k-1)$). This is true for every block of m links on the ring. Figure 4 shows the shortest delay path for nodes within each of k such blocks.

$\forall i : i \in [1,k], \forall j : j \in [1,2]$, let d_{i1} be the delay in block i from node $(i-1)m$ to im, while d_{i2} be the delay in block i from node im to $(i-1)m$ as shown in the figure 4. We claim that $d_{min} = \min_{i,j}\{d_{ij}\} < 2m$. This can again be proved by contradiction as follows:

Consider a path from node 0 to node $\frac{k-1}{2}m$. There are two possibilities as shown in figure 4:

1) $0 \to m \to 2m \cdots \to \frac{k-1}{2}m$. The delay along this path is at least $\frac{k-1}{2}d_{min}$.
2) $0 \to mk-m \cdots \to \frac{k-1}{2}m$. The delay along this path is at least $\frac{k+1}{2}d_{min}$

Thus, if $d_{min} \geq 2m$, it contradicts the assumption that $D_{f'} < m(k-1)$. Moreover, since each block has m links, each incurring a delay of at least 1,

$$m \leq d_{min} < 2m$$

Let $d_{min} = m + x$, where $x \in [0, m)$. Consider the block that has the lowest delay d_{min}. Without loss of generality, label the starting and ending node in this block as $mk-m$ and 0 as shown in the figure 5. Consider a path from node 0 to node $mk-m-x$. There are two possibilities as shown in figure 5:

1) $0 \to mk-m \to \cdots \to mk-m-x$. Delay along this path is at least $mk - d_{min} + x = m(k-1)$, which contradicts our assumption about $D_{f'} < m(k-1)$.
2) $0 \to m \to 2m \cdots \to mk-m-x$. Delay along this path is given by:

$$\begin{aligned} D &\geq \sum_{i=1}^{k-2} d_i + (m-x) \\ &\geq (k-2)(m+x) + m - x \\ &\geq m(k-1) + x(k-3) \\ &\geq m(k-1)(\text{for } \ k \geq 3) \qquad (8) \end{aligned}$$

This again contradicts our assumption that $D_{f'} < m(k-1)$.

Thus, for the ring with $n = mk$ nodes, the sequential assignment minimizes the *delay diameter*. ■

For the case when $n = mk + t$, for $0 < t < k$, the optimal solution is slightly more involved.

Theorem 4: For a ring with n nodes where $n = mk + t$, for $0 < t < k$, the following is a lower bound on the delay diameter:

$$D_f \geq (m+1)k - \lfloor \frac{(m+1)k - y}{x} \rfloor \qquad (9)$$

where $n = mk + t = (m+1)x + y$.

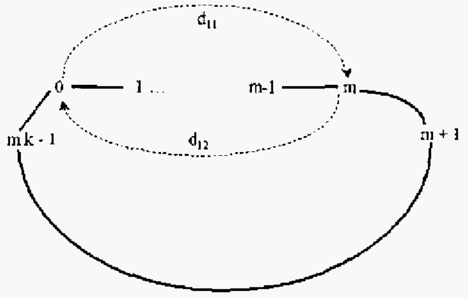

Fig. 3. Shortest delay path for a single block of m links.

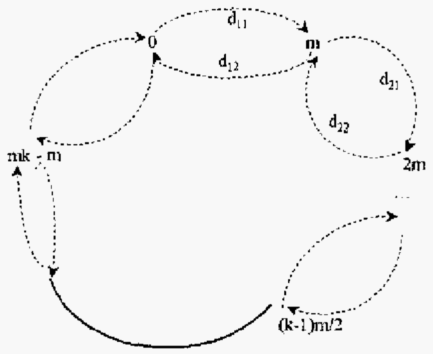

Fig. 4. Shortest delay path for k blocks of m links each.

The proof is described in appendix X. A slot assignment that achieves this lower bound is illustrated by the figure 6 (b).

In section V, we describe some centralized and distributed heuristics for slot assignment on general topologies.

V. Heuristic Approaches

From the theoretical analysis, we know that DESS is NP-hard, hence it is unlikely that there exist polynomial time algorithms for solving it. We instead propose several heuristic solutions in this section and evaluate their performance through simulations in section VI.

A. *Centralized Algorithm*

Initially, all nodes are assigned the same slot and the *delay diameter D* of the network is computed. By either deterministic or random order, each sensor node calculates the

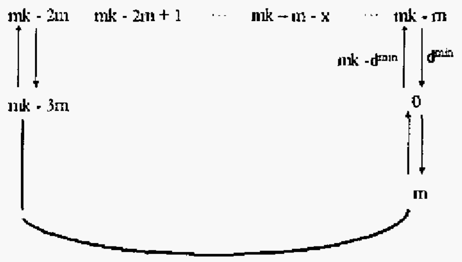

Fig. 5. Paths from node 0 to node $mk - m - x$

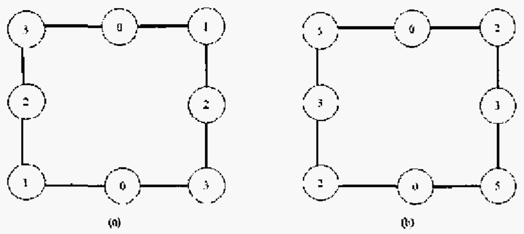

Fig. 6. (a) The sequential slot assignment f obtained for a ring with $n = 8$ nodes and $k = 4$ slots ($n = mk$). Here $D_f = 6$. (b). A slot assignment f obtained for a ring with $n = 8$ nodes with $k = 6$ using the optimal construction for ($n = mk + t$). Here $D_f = 9$ which matches the lower bound in equation 9.

delay diameter of the network for all possible slot assignments for itself while keeping other nodes' slots unchanged. If the minimum of the *delay diameters* of all possible slot assignments d_{min} is smaller than the previous *delay diameter* ($d_{min} < d$), the node changes its slot to the one that gives the minimum *delay diameter* and updates $d \leftarrow d_{min}$. If the *delay diameter* is unchanged, it chooses the new slot or keeps the current slot with equal probability. Otherwise it keeps its current slot unchanged. After all nodes finish this operation, the iteration can be repeated again. The number of iterations depends on limitations on the algorithm duration (which in turn depends upon the size of the network). The pseudo code for the centralized algorithm is shown below.

Algorithm *Centralized*

```
1.  Assign slot 0 to all nodes in G
2.  $d = D(G)$            //delay diameter of G
3.  for $i \leftarrow 1$ to $n$      //number of iterations
4.      for each node $s$ in the network
5.          for $k_1 \leftarrow 0$ to $k-1$      //total slots
6.              $ok \leftarrow slot(s)$
7.              $slot(s) \leftarrow k_1$
8.              $md \leftarrow D(G)$
9.              if $d_{min} < d$
10.               then $d \leftarrow d_{min}$
11.                    $minslot \leftarrow k_1$
12.             if $d_{min} == d$
13.               then $minslot \leftarrow k_1$ with 50% probability
14.                    $minslot \leftarrow ok$ with 50% probability
15.             $slot(s) \leftarrow minslot$
```

B. *Localized Algorithms*

The centralized algorithm assumes complete knowledge of the network topology and slot assignment. In this section we consider some localized algorithms in which a sensor node only knows the information stored at its neighbors.

We propose two different localized algorithms.

- Local-Neighbor: A node knows only the slot assignments of its neighbors. It chooses a slot which minimizes the maximum of its delays to and from its immediate neighbors. This can be repeated for a certain number of iterations.
- Local-DV: Its working is similar to Distance Vector routing techniques. Each node maintains two Distance Vector tables: a forward table FDV which stores its shortest delays *to* all other nodes and a backward table BDV which stores its shortest delays *from* all other nodes. These two tables can be calculated using the basic Bellman-Ford technique. A sensor node also knows the DV tables of its direct neighbors. A sensor node calculates the DV tables for all possible new slot assignments for itself. Let the maximum value of entries in the sets of the two DV tables over all possible slot assignments be $maxd$. The node will choose the slot which gives the minimum $maxd$.

The pseudo codes of Local-Neighbor and Local-DV are shown below.

Algorithm *Local-Neighbor*

```
1.  Each node $s$ get the slots of its direct neighbor $N(s)$
2.  $mind \leftarrow MAX\_VALUE$
3.  for $k_1 \leftarrow 0$ to $k-1$      //total slots
4.      $slot(s) \leftarrow k_1$
5.      $fd(s,t) \leftarrow$ delay from $s$ to $t$ in $N(s)$
6.      $bd(s,t) \leftarrow$ delay from $t$ in $N(s)$ to $s$
7.      $maxd \leftarrow max(fd, bd)$
8.      if $maxd < mind$
9.        then $mind \leftarrow maxd$
10.            $minslot \leftarrow k_1$
11.     $slot(s) \leftarrow minslot$
```

Algorithm *Local-DV*

```
1.  Each node $s$ calculate DV tables $FDV, BDV$
2.  Get the $FDV, BDV$ of its direct neighbor $N(s)$
3.  $mind \leftarrow MAX\_VALUE$
4.  for $k_1 \leftarrow 0$ to $k-1$      //total slots
5.      $slot(s) \leftarrow k$
6.      update $FDV, BDV$
7.      $maxd \leftarrow max(FDV, BDV)$
8.      if $maxd < mind$
9.        then $mind \leftarrow maxd$
10.            $minslot \leftarrow k_1$
11.     $slot(s) \leftarrow minslot$
```

C. *Randomization*

The simplest slot assignment is to just randomly choose a slot for each node once. In a dense network where a node has a large number of neighbors (where multiple paths are available for any pair of nodes), there is a high probability that this assignment may lead to a short delay path. We call this decentralized random slot assignment as Random-Average. The performance of this method is evaluated by the expected value of the *delay diameter*. The randomized slot assignment can also be done in a centralized manner. We refer to this centralized version as the Random-Minimum strategy. After a certain number of iterations of choosing random slots for all the nodes, this strategy chooses the assignment that gives the minimum *delay diameter* and then deploys the slot assignment in the network.

While all the above heuristics can be used for any topology, we next propose a specialized heuristic for the grid that

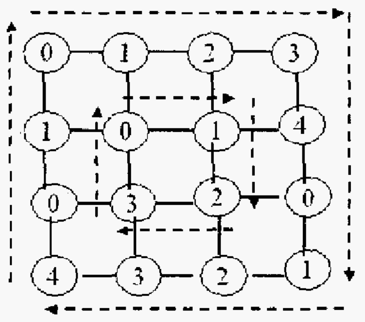

Fig. 7. Concentric ring allocation for a grid of 4×4 nodes with $k = 5$. The dotted lines illustrate the concentric rings at each level.

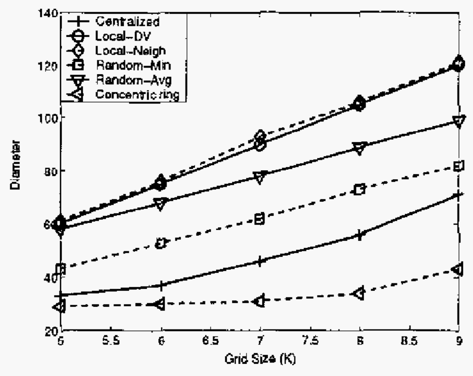

Fig. 8. The *delay diameter* of the heuristic algorithms versus grid size for the number of slots fixed at $k = 15$. The grid is given as $X \times X$.

exploits the structure of the topology.

D. Concentric Ring for the Grid topology

We believe that the optimal assignment on a ring can serve as a basis for a low latency assignment on a grid that can be viewed as a set of concentric rings with interconnecting bridges. The outer most ring is given a sequential assignment going in the clock-wise direction starting at 0. For every other ring, a slot assignment is chosen that offers the best *delay diameter* for that ring. An example of this assignment is shown in figure 7

VI. Simulation Results

In this section, we evaluate the performance of the heuristic algorithms on the grid topology (in section VI-A) and random topology (in section VI-B) through high level simulations. Since the current study focuses on comparing the *delay diameter* only across these heuristics, even the distributed algorithms are simulated in a centralized manner (without analyzing their overhead). We also assume that the number of slots k is dictated by the duty cycling requirements of the application.

A. Grid Network

First we evaluated the six schemes on a grid topology: Centralized, Local-DV, Local-Neighbor, Random-Avg, Random-Min and Concentric Ring. To have a fair comparison, the centralized and the two local schemes had the same number of iteration $I = 20$ while the random algorithms ran for $I \times k$ times.

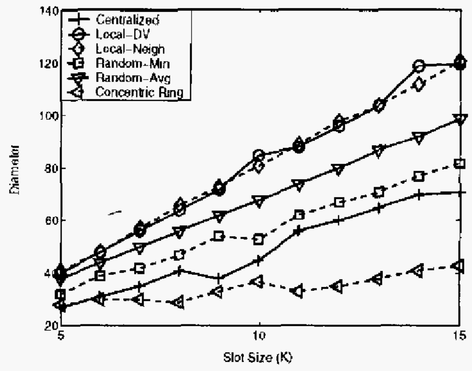

Fig. 9. The *delay diameter* of the heuristic algorithms versus the number of slots (k) for a fixed grid size of 9×9.

Figure 8 shows the results for different grid sizes while the number of slots k is fixed at 15. By exploiting the structure of the grid, concentric ring has the best performance compared to all other schemes. The centralized scheme is slightly worse than the concentric ring at small grid size but is about 2 times worse than concentric ring when grid size is large. Both the randomized schemes perform worse than the centralized algorithm with Random-Min doing better than Random-Avg. Moreover the two localized algorithms also seem to perform poorly compared to the centralized one. This is possibly because of the fact that the *delay diameter* of a network being a global property, the local optimization schemes do not converge to the global optimum. Overall, we find that the centralized scheme can reduce the *delay diameter* of random schemes by about 50%, while the concentric ring can provide a further reduction of about 50%.

Although k should be decided by the duty cycle requirement of applications, it is interesting to see its impact on the *delay diameter*. Figure 9 shows the results for different values of k while the grid size is fixed at 9×9. Clearly, the *delay diameter* increases almost linearly with the number of slots k. Concentric ring performs the best while local schemes perform the worst. We further evaluated these schemes on a larger grid with 20×20 nodes and values of k up to 20. We observed similar trends in performance.

B. Random Network

We also tested the five schemes (excluding the Concentric Ring heuristic) on a network with randomly deployed sensor nodes.

First we fixed the radio transmission range at 2. Figure 10 and 11 show the result with 100 nodes uniformly distributed in a 10×10 square and a 3×33 rectangle. In both cases, the centralized scheme performs best, followed by Random-Min. It is interesting to note that in the random network, Local-Neighbor now has a smaller *delay diameter* than Random-Avg. In the 3×33 area, the Local-Neighbor performs quite well even on comparison with the Random-Min scheme. We believe this is not because Local-Neighbor perform better but because

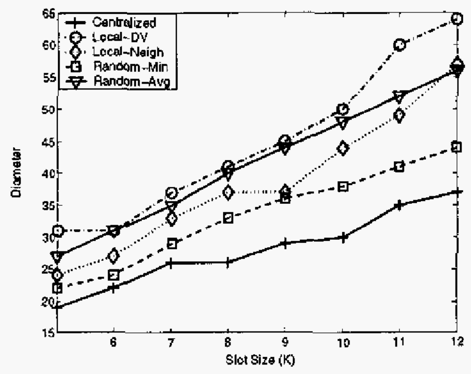

Fig. 10. The *delay diameter* of the heuristic algorithms versus the number of slots (k) for nodes randomly deployed in a 10×10 area. The transmission range is 2.

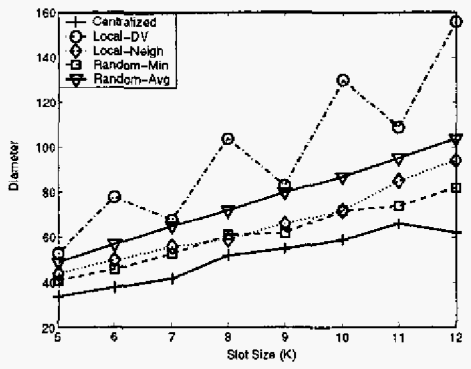

Fig. 11. The *delay diameter* of the heuristic algorithms versus the number of slots (k) for nodes randomly deployed in a 3×33 area. The transmission range is 2.

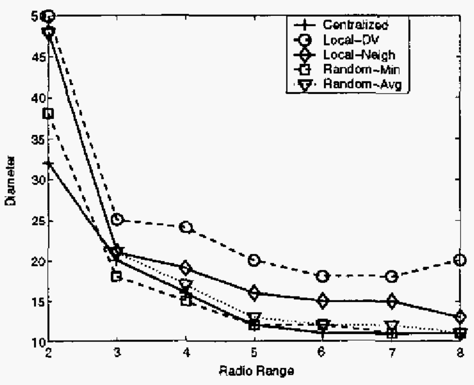

Fig. 12. The *delay diameter* of the heuristic algorithms versus the radio transmission range for nodes randomly deployed in a 10×10 area. Number of slots is fixed at $k = 10$.

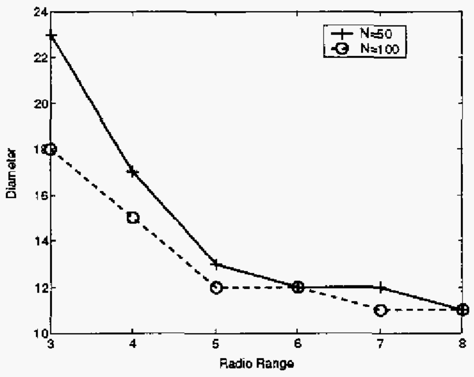

Fig. 13. The *delay diameter* of the Random-Min algorithm versus radio transmission range for nodes randomly deployed in a 10×10 area with $N = 50$ and $N = 100$. The number of slots $k = 10$.

Random schemes perform worse in a random graph. In a grid, each internal node has 4 direct neighbors. In a random graph, however there is a probability that some nodes are bottlenecks (nodes through which several paths go through). An improper slot assignment for such a bridge node may hurt the *delay diameter* significantly. In a 3×33 long rectangle, the probability of a node being a bridge becomes higher, which is likely the reason that the performance of Local-Neighbor is closer to the Random-Min scheme. This intuition is also backed by figure 13 which shows the *delay diameter* obtained by the random schemes with 50 and 100 nodes. With 100 nodes, the density and the average degree of the network increases, the random schemes have better performance because of the increased number of paths between any pair of nodes (and hence fewer bottlenecks).

Figure 12 shows the effect of the radio transmission range R on the *delay diameter*. As R increases, the *delay diameter* decreases. This is because an increase in R decreases the graph diameter (in hops).

Thus, for the single schedule case, where each node chooses exactly one of the k slots to wake up, we have presented several heuristics in section V and evaluated them through simulations in section VI. In section VII, we show that by carefully choosing multiple wake up slots for each sensor, we gain significant delay savings over the single schedule case at the same duty cycling. Using the multi-schedule technique, we propose algorithms with provable delay guarantees.

VII. Multi-schedule Solutions

In this section, we show how significant savings in terms of delay can be obtained if we use multi-scheduling where we allow sensors to wake up at multiple time slots. The schedule length is increased proportionately so that each sensor remains active for only $\frac{1}{k}$ fraction of the time slots on an average. We will first demonstrate simple multi-schedules for tree and grid networks where the latency between two sensors is at most $d+O(k)$ where d denotes the shortest path distance (in hops) between the sensors in the original network. This is a very useful guarantee, since this holds not just for the diameter-pair but for an arbitrary pair of sensors. The difference between the shortest path and the latency of our algorithm is only an additive $O(k)$, which is independent of the network size and the distance between the communicating sensors. We obtain weaker a guarantee for general networks.

A. Tree Networks

Suppose the sensor network forms a tree. Arbitrarily choose a sensor r as the root of the tree, and denote $l(X)$ as the shortest path distance of sensor X (in hops) from r.

Consider the multi-schedule where X is awake during time slot t iff either $t - l(X)$ or $t + l(X)$ is divisible by $2k$. Thus a sensor is only awake for at most a $\frac{1}{k}$ fraction of the time slots. The multi-schedule length is $2k$.

Consider two sensors X and Y and let Z denote the least-common-ancestor of X and Y in the rooted tree. Then the shortest path distance d between X and Y is given by $(l(X) - l(Z)) + (l(Y) - l(Z))$; this corresponds to going up from X to Z and then down from Z to Y. Informally, each sensor is a part of two synchronized schedules, one going up the tree and the other going down. Hence we will call our algorithm TREE-MULTI-SYNCH. Assume that sensor X gets a packet at time t. To send a packet from X to Y, the algorithm first waits for time t_1 such that $t_1 + l(X) \bmod 2k = 0$. Then, this packet is transmitted from X to Z, one hop in one time slot, reaching Z at time t_2. The algorithm then waits for time t_3 such that $t(3) - l(Z) \bmod 2k = 0$, and then the packet is transmitted from Z down to Y, one hop in one time slot, till it reaches Y at time t_4. The total latency for the packet is $t_4 - t = (t_1 - t) + (t_2 - t_1) + (t_3 - t_2) + (t_4 - t_3)$. Now, $(t_4 - t_3) + (t_2 - t_1) = d$. Further, at least one of the numbers $t, t+1, t+2, \ldots, t+2k-1$ is divisible by $2k$. Hence, $t_1 - t \leq 2k - 1$. Similarly, $t_3 - t_2 \leq 2k - 1$, yielding the following theorem:

***Theorem** 5:* The total latency (and hence the *delay diameter*) due to the TREE-MULTI-SYNCH algorithm is less than $d + 4k$.

Note that this is significantly better than the lower bound of $\Omega(dk)$ on the latency for the single wake up schedule (obtained in section IV-B.1), and hence multi-schedules are *provably* better than single wake up schedules for latency.

B. Grid Networks

The multi-schedule is very simple. Consider sensor X at position (i, j). X is awake during time slot t iff at least one out of $t + i$, $t - i$, $t + j$, and $t - j$ is divisible by $4k$. During any interval of length $4k$, a sensor is awake for at most 4 time slots. Hence, on an average, a sensor is awake for at most a $\frac{1}{k}$ fraction of the time slots. The multi-schedule length is $4k$.

We will now describe the algorithm for transferring a packet of information from sensor X at position (i, j) to sensor Y at position (p, q). Informally, each sensor is a part of four synchronized schedules, one for each of the directions, and hence we will call our algorithm GRID-MULTI-SYNCH.

Without loss of generality, assume $i \leq p$ and $j \leq q$. Suppose the packet becomes available at time t at sensor X. Let $t_1 \geq t$ denote the first time instant such that $t_1 - i \bmod 4k = 0$. For all r, $1 \leq r \leq p - i - 1$, the sensor $(i + r, j)$ transmits the packet at time $t_1 + r$. Since $t_1 - i \bmod 4k = 0$, it follows that $((t_1 + r) - (i + r)) \bmod 4k$ must also be 0. Hence, sensor $(i + r, j)$ is awake at time $t_1 + r$ and can receive this packet in time to transmit it during the next time slot $t_1 + r + 1$. The packet arrives at sensor (p, j) at time $t_2 = t_1 + p - i$.

Let $t_3 \geq t_2$ denote the smallest time such that $t_3 - j \bmod 4k = 0$. For all r, $1 \leq r \leq q - j - 1$, sensor $(p, j + r)$ transmits during time slot $t_3 + r$. Using the same reasoning as above, the packet arrives at sensor (p, q) at time $t_4 = t_3 + q - j$.

***Theorem** 6:* Let $d = (p - i) + (q - j)$ denote the shortest path distance between X and Y. Then the total latency (and hence the *delay diameter*) due to the GRID-MULTI-SYNCH algorithm is less than $d + 8k$.

Proof: The total latency for the packet is $t_4 - t = (t_4 - t_3) + (t_3 - t_2) + (t_2 - t_1) + (t_1 - t)$. But $t_4 - t_3 = q - j$ and $t_2 - t_1 = p - i$. Hence, $(t_4 - t_3) + (t_2 - t_1) = p - i + q - j = d$. Further, at least one of the numbers $t, t+1, t+2, \ldots, t+4k-1$ is divisible by $4k$, and hence $t_1 - t \leq 4k - 1$. Similarly, $t_3 - t_2 \leq 4k - 1$. Thus, the latency of the above algorithm is at most $d + 8k - 2$. ■

C. General Networks

We will use a general result about decomposition of networks into trees. Suppose we are given an unweighted graph G over n nodes. Let $d_G(u, v)$ denote the shortest path distance between nodes u and v in G. Let c be a large enough constant. The following result is implicit in the work of Bartal [21], [22] and Fakcheroenphol, Rao, and Talwar [23]:

***Theorem** 7:* ([21], [22], [23]) There exists a collection S of $c \log n$ spanning trees of G such that for all nodes u, v in G,

$$\min_{T \in S} d_T(u, v) \leq d_G(u, v) \cdot c \log n.$$

Let $l_T(X)$ denote the level (i.e. distance from the root) of sensor X in tree $T \in S$. Sensor X is alive at all times t such that either $t - l_T(X)$ or $t + l_T(X)$ is divisible by $(2k)(c \log n)$ for some $T \in S$. The multi-schedule length is now $2ck \log n$, and each sensor is awake for at most a fraction $\frac{1}{k}$ of the time slots. To send a packet of information from X to Y, we will find the tree $T \in S$ which minimizes $d_T(X, Y)$, and use the TREE-MULTI-SYNCH algorithm on T. Since the multi-schedule length is $2ck \log n$, the latency (and hence the *delay diameter*) due to this algorithm will be at most $d_T(X, Y) + 4ck \log n = O((d_G(X, Y) + k) \log n)$.

VIII. CONCLUSIONS AND FUTURE WORK

In this paper we have addressed the important problem of minimizing communication latency while providing energy-efficient periodic sleep cycles for nodes in wireless sensor networks. The objective is to minimize the latency given the duty cycling requirement that each sensor has to be awake for $\frac{1}{k}$ fraction of time slots on an average. For the single wake up schedule case, where each sensor can wake up at exactly one of the k slots, we have provided graph-theoretic problem formulations for arbitrary all-to-all (DESS) as well as weighted communication patterns (ADESS). We also proved that both these problems are NP-hard. We then focused on the DESS problem and derived and proved optimal solutions for two special cases, *viz.* the tree and ring topologies. For arbitrary topologies, we proposed several heuristics and evaluated them through simulations. These simulations reveal several interesting observations: that purely localized heuristics tend to perform worse than simple randomized slot allocations, that our centralized scheme can provide delay reductions of around 50% over randomized schemes and that specialized heuristics (that exploit the topological structure) like the concentric ring for the grid can provide additional gains. Further, we showed that by carefully choosing multiple wake up slots, one can obtain significant savings in the latency at the same duty cycling. Using this technique, we propose algorithms with provable guarantees on tree, grid and arbitrary graphs. These results obtained from an algorithmic perspective are novel and quite different from prior work in this area which has focused primarily on intuitive MAC protocol designs (such as S-MAC [1], T-MAC [3], and our own work on D-MAC [10]). In many ways we have only discovered the tip of an iceberg in this domain. Many interesting and challenging open problems arise that we would like to pursue in our own future work and present to the research community:

- Techniques to compute good lower bounds on the optimal *delay diameter* for an arbitrary graph.
- Good distributed heuristics for the DESS problem.
- In-depth analysis and algorithms for the weighted communication average delay problem (ADESS).
- Incorporation of local interference constraints similar to TDMA scheduling problems to handle moderate to high traffic scenarios.
- Implementation and validation of delay efficient schedules in testbed/real-world sensor networks.

IX. Acknowledgement

The authors wish to thank the reviewers for their invaluable comments. Bhaskar Krishnamachari would like to thank NSF for partially supporting his research under grants 0435505, 0325875, 0430061, and CAREER award 0347621. Ashish Goel would like to thank NSF (grants 0428868, an NSF CAREER award 0339262) and the Alfred P. Sloan Foundation (Faculty Fellowship) for partially supporting his research.

References

[1] Wei Ye, John Heidemann, and Deborah Estrin, "An Energy-Efficient MAC protocol for Wireless Sensor Networks," in *IEEE Infocom*, 2002.
[2] W. Ye, J. Heidemann, and D. Estrin, "Medium Access Control with Coordinated, Adaptive Sleeping for Wireless Sensor Networks", Technical Report USC ISI-TR-567, January, 2003. (Accepted to appear in *ACM/IEEE Transactions on Networking*.)
[3] Tijs van Dam, Koen Langendoen, "An Adaptive Energy-Efficient MAC Protocol for Wireless Sensor Networks", in *ACM Sensys* Nov. 2003.
[4] Rong Zheng, Robin Kravets, "On-demand Power Management for Ad Hoc Networks", in *IEEE Infocom* 2003.
[5] Rong Zheng, Jennifer C. Hou and Lui Sha, "Asynchronous Wakeup For Ad Hoc Networks", in *ACM MobiHoc* 2003.
[6] Eun-Sun Jung, Nitin H. Vaidya, "An Energy Efficient MAC Protocol for Wireless LANs", in *IEEE Infocom* 2002.
[7] C. S. Raghavendra and S. Singh, "PAMAS-power aware multi-access protocol with signaling for ad hoc networks", in *Computer Communication Review*, 1998.
[8] A. El-Hoiydi, J. D. Decotignie, C. Enz and E. Le Roux, "WiseMAC: an Ultra Low Power MAC Protocol for the WiseNET Wireless Sensor Network", Poster, in *ACM Sensys* 2003.
[9] O. Dousse, P. Mannersalo, P. Thiran, "Latency of Wireless Sensor Networks with Uncoordinated Power Saving Mechanisms", *MOBIHOC* 2004.
[10] Gang Lu, Bhaskar Krishnamachari, Cauligi Raghavendra, "An Adaptive Energy-Efficient and Low-Latency MAC for Data Gathering in Sensor Networks", in *4th IEEE International Workshop on Algorithms for Wireless, Mobile, Ad Hoc and Sensor Networks WMAN*, April 2004.
[11] Michael R. Garey and David S. Johnson, "Computers and Instractability: A Guide to the Theory of NP-Completeness. W.H Freeman, San Francisco", CA, 1979.
[12] W. Liang, "Constructing Minimum Energy Broadcast Trees in Wireless Ad Hoc Networks", *MOBIHOC* 2002.
[13] Jeremy Elson, Lewis Girod and Deborah Estrin, "Fine-Grained Network Time Synchronization using Reference Broadcasts", in *ACM SIGOPS* 2002.
[14] J. Gruenen and J. Rabaey, "Lightweight time synchronization for sensor networks", in *ACM WSNA*, 2003.
[15] S. Ganeriwal, R. Kumar, and M.B. Srivastava, "Timing-Sync Protocol for Sensor Networks", in *ACM SenSys*, 2003.
[16] L. C. Pond and V. O. K. Li, "A distributed time-slot assignment protocol for mobile multi-hop broadcast packet radio networks", *MILCOM*, pp. 70-74, 1989.
[17] D. Ganesan, B. Krishnamachari, A. Woo, D. Culler, D. Estrin and S. Wicker, "Complex Behavior at Scale: An Experimental Study of Low-Power Wireless Sensor Networks", UCLA CS Technical Report UCLA/CSD-TR 02-0013, 2002.
[18] J. Zhao and R. Govindan, "Understanding Packet Delivery Performance in Dense Wireless Sensor Networks", *ACM Sensys*, November 2003.
[19] A. Woo, T. Tong, and D. Culler, "Taming the Underlying Issues for Reliable Multhop Routing in Sensor Networks", *ACM SenSys*, November 2003.
[20] A. Cerpa, N. Busek, and D. Estrin, "SCALE: A tool for Simple Connectivity Assessment in Lossy Environments", CENS Technical Report, September 2003.
[21] Y. Bartal, "Probabilistic approximation of metric spaces and its algorithmic applications", *37th IEEE Symposium on Foundations of Computer Science*, pages 184–193, 1996.
[22] Y. Bartal, "On approximating arbitrary metrics by tree metrics", *30th ACM Symposium on Theory of Computing*, 1998.
[23] J. Fakcheroenphol, S. Rao, and K. Talwar, "A tight bound on approximating arbitrary metrics by tree metrics", *Proceedings of ACM Symposium on Theory of Computing*, 2003.

X. Appendix

***Lemma** 1:* Consider $n = mk + t$, $0 < t < k$. $C_1 = \sum_i d_f(i, i+1) + d_f(n-1, 0) = M \cdot k$, where $i \in [0, n-1]$, $M \geq m+1$.

Proof:

Since, there are $mk + t$ links and the delay on each link under any slot assignment is at least 1, M has to be at least $m + 1$. Now,

$$d_f(i,j) = (S_j - S_i) \bmod k = \begin{cases} S_j - S_i & \text{if } S_j - S_i > 0 \\ S_j - S_i + k & \text{if } S_j - S_i \leq 0 \end{cases}$$

So:

$$\begin{aligned} C_1 &= \sum_i d_f(i, i+1) + d_f(n-1, 0) \\ &= (S_1 - S_0) \bmod k + \ldots + (S_0 - S_{n-1}) \bmod k \\ &= M \cdot k + (S_1 - S_0 + S_2 - S_1 + \ldots + S_0 - S_{n-1}) \\ &= M \cdot k \end{aligned}$$

It is easy to know that $C_2 = \sum_i d_f(i, i-1) + d_f(0, n-1) = M \cdot k$, where $i \in [1, n]$, $M \geq m+1$. $C_1 + C_2 = (mk+t)k$. Without loss of generality, let $C_1 \leq C_2$, then $M \leq \frac{mk+t}{2}$. ■

***Lemma** 2:* For an optimal slot assignment, $M = m + 1$.

Proof: It can be shown that the sequential slot assignment which assigns a sequentially increasing slot (by one and modulo k) has a *delay diameter* of $(m+1)(k-1)$. We will show that for $M > m+1$, the *delay diameter* will always be larger than $(m+1)(k-1)$. Assume $M = m+2$, hence $C_1 = (m+2)k$. We break the ring into blocks of size $m+2$:

$$0 \rightarrow 1 \rightarrow 2 \ldots m+1 \rightarrow m+2$$

$$1 \rightarrow 2 \rightarrow 3 \ldots m+2 \rightarrow m+3$$

$$\vdots$$

$$mk+t-1 \rightarrow 0 \rightarrow 1 \ldots m \rightarrow m+1$$

Let d_i be the sum of the all the link delays of the block starting at node i.

$$\begin{aligned}\sum_{i=0}^{mk+t-1} d_i &= (m+2)\cdot C_1 \\ &= (m+2)(m+2)k\end{aligned}$$

Let d_{min} be the minimum of all d_is, thus:

$$\begin{aligned}(mk+t)d_{min} &\leq (m+2)(m+2)k \\ d_{min} &\leq \frac{(m+2)(m+2)k}{mk+t} \\ d_{min} &\leq m+2+\frac{(2k-t)(m+2)}{mk+t}\end{aligned}$$

Consider the block that has the lowest delay d_{min}. Without loss of generality, let $d_0 = d_{min} = m+2+x$ shown in figure 14, where $x = \lfloor\frac{(2k-t)(m+2)}{mk+t}\rfloor$. Consider the path from node $m+2$ to node 0. There are two possibilities:

1) $m+2 \rightarrow m+3 \cdots \rightarrow mk+t-1 \rightarrow 0$. The delay along this path is $(m+2)k - d_0$.
2) $m+2 \rightarrow m+1 \cdots \rightarrow 1 \rightarrow 0$. The delay along this path is also $(m+2)k - d_0$.

For both case, the delay D is given by:

$$\begin{aligned}D &= (m+2)k - d_0 \\ &= (m+2)k-(m+2+x) \\ &= (m+1)(k-1)+k-1-x\end{aligned}$$

Since $M \leq \frac{mk+t}{2}$, when $M = m+2$:

$$\frac{m+2}{mk+t} \leq \frac{1}{2}$$

Also because $0 < t < k$ and $k \geq 3$:

$$\lfloor\frac{2k-t}{k-1}\rfloor \leq 2$$

So:

$$\begin{aligned}\lfloor\frac{m+2}{mk+t}\cdot\frac{2k-t}{k-1}\rfloor &\leq 1 \\ x = \lfloor\frac{2k-t}{mk+t}(m+2)\rfloor &\leq k-1 \\ k-1-x &\geq 0 \\ D &\geq (m+1)(k-1)\end{aligned}$$

Thus we have proved that when $C_1 = (m+2)k$, the *delay diameter* will be at least $(m+1)(k-1)$. Similarly, it can be proved that for any $M \geq (m+2)$, the *delay diameter* will be no smaller than $(m+1)(k-1)$.

Hence for an optimal slot assignment, $C_1 = (m+1)k$. ∎

Now we will calculate the lower bound on the *delay diameter* of the ring when $n = mk+t$. Similarly as the case when $n = mk$, we break the ring into blocks of size $m+1$ shown in figure 15.

$$n = mk+t = (m+1)x+y \tag{10}$$

where $0 \leq y < m+1$

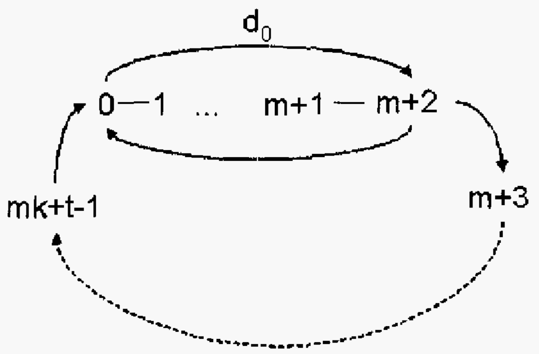

Fig. 14. Paths from node $m+2$ to node 0

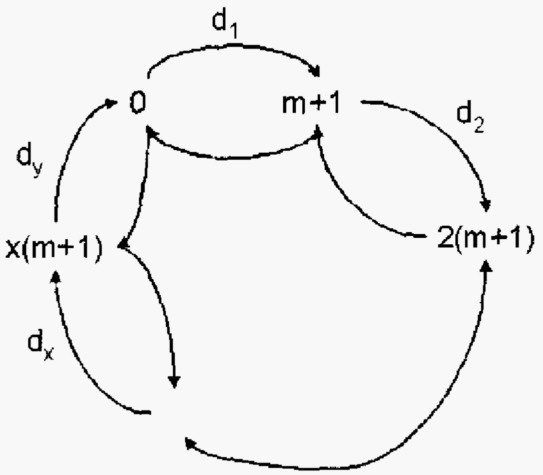

Fig. 15. Shortest delay for x blocks of $m+1$ links each

For any possible such block of $m+1$ links, let d_{min} be the minimum delay. The *delay diameter* of the ring is $(m+1)k - d_{min}$. If we get the maximum value of d_{min}, we then achieve the smallest diameter $D = (m+1)k - \max(d_{min})$.

Since $\sum d_i = (m+1)k$, we have:

$$\begin{aligned}x\cdot d_{min}+d_y &\leq (m+1)k \\ d_{min} &\leq \frac{(m+1)k-d_y}{x} \leq \frac{(m+1)k-y}{x} \\ \max(d_{min}) &= \lfloor\frac{(m+1)k-y}{x}\rfloor\end{aligned}$$

Thus, the lower bound on the *delay diameter* D_f for any slot assignment function f is given by:

$$D_f \geq (m+1)k - \lfloor\frac{(m+1)k-y}{x}\rfloor$$