

# Efficient Scheduling for Sensor Networks

Andreea Berfield and Daniel Mossé

Department of Computer Science, University of Pittsburgh

{andreea, mosse}@cs.pitt.edu

**Abstract**—Sensor networks opened new opportunities to monitor the environment. In order to retrieve the desired data, sensors are usually organized into a hierarchy and synchronize when transmitting the data towards the base station. Many scheduling schemes have been proposed with the goal of allowing sensors to sleep as much as possible and ultimately save energy. In this paper, we propose two new scheduling algorithms that assign predefined slots to each sensor. These algorithms are distributed, need very little global information and do not need knowledge about the location of sensors or the network topology. As others, we also assume that loose clock synchronization is available.

The experimental results confirm our expectations. They show a significant reduction in the average time awake per node of at least three times compared to more traditional routing protocols like TAG.

## I. INTRODUCTION AND RELATED WORK

Sensor technology has received tremendous attention in the last years. This increasing interest is due to the benefits sensors bring and the large scale of applications previously unexplored. However, the challenges sensor networks face limit their usability. Among the most notable are: limited energy, limited communication and processing capacity, failures of sensors and inaccessibility in most situations.

Given such a challenging environment, sensors should be programmed efficiently so that long-living functional sensor networks are ensured. In order to achieve this goal, sensors synchronize their transmission schedule to allow them to sleep longer and thus conserve their energy.

One solution to obtain minimum average time awake per node is to employ a time division multiple access (TDMA) scheduling technique. Many such existing scheduling schemes use graph coloring algorithms that need global information to compute the transmission schedule for the entire network. Afterwards, they distribute the schedule to the sensors. This process requires the exchange of many large messages and also has the disadvantage of a central point of failure. There are also distributed versions that only require knowledge of the neighborhood within one or two hops ([8], [17]).

Another graph-based coloring algorithm for determining the initial time slot of each sensor requires sensors to be aware of only their immediate neighbors, not of the whole network topology [2]. Synchronization is achieved by token circulation. Whenever a sensor receives the token, it can synchronize its clock with respect to its parent, choose a color from the available colors set, update its neighbors about its current color and forward the token. The initial

problem is transformed into a graph-coloring problem, which is a NP-problem, but can be solved with approximation algorithms. Based on the initial color and the token circulation period, the node can determine all its time slots. Before forwarding the token, the sensor reports its color to its distance-two neighborhood. Both the sensor and its distance-one neighbors broadcast this color. Therefore, there are at most  $d + 1$  update messages before each sensor forwards the token, where  $d$  is the maximum degree of nodes in the graph. Periodically the base station (BS) checks whether or not the token was lost and if needed, it initiates the recovery process. The recovery process is usually three times longer than the token circulation time. The algorithm achieves self-stabilization and robustness to failures by requiring a high number of messages exchanged for the initialization and recovery phases (at least  $O(dN)$  messages, where  $N$  is the number of nodes in the tree).

A self-repairing, leader-assigned, distance-two coloring algorithm is presented in [11]. It reacts to failures and dynamic network topologies. It uses a randomized slot assignment algorithm that is expected to achieve local stabilization in  $O(1)$ , but it does not guarantee global stabilization. The algorithm uniquely identifies neighborhoods of nodes and elects a leader that will decide the coloring for all the nodes in its neighborhood. The TDMA schedule is composed from a series of slots where sensors transmit their messages and a fixed-length slot, called overhead. The overhead is reserved for messages of the algorithm to assign colors and time slots to nodes for the next round of messages. The algorithm performs graph coloring or schedule construction every epoch. The main disadvantage of this approach is that a single node joining or disappearing from the network can disrupt the TDMA schedule of a sizable part of the network before the system stabilizes.

The authors of [5] propose a sensor scheduling protocol that guarantees a bounded-delay sensing coverage and maximizes the network lifetime. Initially, each node randomly selects a wakeup time and communicates it to its neighbors. In the following iterations, each node recalculates its wakeup time exactly once per iteration, based on the most recently updated neighbor schedules. The wakeup time is finalized whenever a node does not receive any update within an iteration and does not change its wakeup time. Eventually all nodes reach a local scheduling decision. This protocol solves a problem orthogonal to ours, because it was designed for rare-event detection and not for continuously reporting readings from the sensor network.

The protocol presented in [12] is adaptive and uses local information. A node chooses its slot by negotiating it with its parent. The two-level architecture combines coarse grain scheduling at the routing layer to plan radio on/off times and fine grain medium access control to provide channel access. However, unlike our work, the protocol does not take into account the “hidden terminal” problem.

Different query plan optimization approaches are discussed in [4]. The distributed query execution plan creates a tree structure for each execution plan. The paper also mentions two options for hierarchical scheduling of operators: a conservative (top-down) approach and a liberal (bottom-up) approach, but does not go into details of how to optimally implement them. The problem of scheduling the operators in the query plan can be viewed as similar to the problem of scheduling the sensor transmissions. The two hierarchical scheduling approaches in [4] represent the inspiration for our sensor transmission scheduling.

Another source of inspiration for our proposed solutions is [6]. This paper presents an event-based communication model for a network organized as a dissemination tree. It assumes a publish/subscribe model, where nodes subscribe to different events. The scheduler relies on topology information to create the schedule. In the Topology-Divided Dynamic Event Scheduling protocol (TD-DES), the root of the tree creates the data dissemination schedule and propagates it to the sensors every iteration (deterministic scheduling) or periodically (speculative scheduling). The paper explains in detail how to distribute the schedule through the network, but not how to create the schedule.

Finally, a data transmission algebra (DTA) was developed in [20]. The DTA helps the query optimizer to generate query routing trees that maximize collision free concurrent data transmissions for a given query or data acquisition model. As with all coverage-based schemes, the query optimizer needs to know the network topology, the applications coverage requirements and the collision domains of the sensor nodes. The DTA consists of a set of operations that take transmissions between wireless sensor nodes as input and produce a schedule of transmissions as their result. This schedule is disseminated by the BS to all the sensors in the query evaluation tree. The potential scalability problem can be addressed by pruning unpromising paths in the search space. Obtaining knowledge about the collision domains is time consuming in a large network. DTA was extended in [16] as a whirlpool, which divides the network into sectors and utilizes an intra-sector concurrency. The paper shows the tradeoffs involved with different whirlpool sectoring and rotation speed. Providing the means to determine the optimal choice represents future work. As part of our future work, we intend to solve a similar problem of scheduling sub-trees in parallel.

In this paper, we propose distributed algorithms that dynamically create the sensor schedule and propagate it throughout the network. These algorithms need only a small

amount of global knowledge, as well as a small number and size of messages. The resulting schedule is similar to the TDMA protocol because it defines fixed time slots for each sensor. However, the means to achieve the schedule and the efficiency of the schedule itself differentiate the algorithms from previous TDMA protocols.

The remainder of the paper is organized as follows: Section II presents our sensor scheduling algorithms, followed by examples in Section III. Section IV contains the experimental results, Section V presents possible issues and optimizations to proposed protocols and describes the future work, and finally Section VI summarizes our conclusions.

## II. SCHEDULING PROTOCOLS

TAG [13] and Cougar [19] are the main non-TDMA protocols used for routing in sensor networks. They use collision-detection and retransmission in order to communicate. In the original Cougar protocol, nodes are awake all the time; they send within a fixed time interval, after which the nodes are in listening mode. Alternatively, the TAG protocol saves energy by keeping nodes from level  $i$  in the tree hierarchy active for receiving from level  $i+1$  and for sending to level  $i-1$ , after which they go to sleep (Fig. 1). The receiving interval for a sensor node is big enough to include receiving the messages from its children, sensing its own data and performing any additional processing/aggregation of information.



Fig. 1. Tag Protocol Synchronization

On the other hand, TDMA-like scheduling has numerous advantages, including faster response, collision-avoidance and ultimately energy savings. However, as already mentioned, most of existing TDMA-like scheduling protocols need global knowledge about the sensor network or exchange too many messages. In this paper, we address these limitations and we propose two efficient scheduling protocols, called *ETDMA* (Efficient TDMA) for practical systems, and *OTAG* (Optimal TAG) for baseline comparison, with further assumptions like timers and clock synchronization. These protocols create a TDMA-like transmission schedule that maximizes the average amount of sleeping time for sensors and thus saves energy.

Any algorithm for querying a sensor network needs at least one initial pass (from the BS) to allow sensors to organize into a hierarchical structure (e.g., Cougar). Most of the algorithms, such as TAG, need two initial passes. The second pass (to the BS) is for informing the parent nodes

which nodes are their children. Both our solutions require one extra pass before the network can start transmitting readings to the user, with complexity  $O(N)$ . In the first pass, the BS broadcasts a message to create the routing tree. Each sensor that hears it selects the sender of the message as its parent and further broadcasts the message. This pass is over when all the sensors have selected a parent in the routing tree. In the second pass, the sensors inform their parents about the time necessary for the sensors in their subtree to finish work and transmit their results, as shown in Algorithm 3. Finally, in the last round, the BS propagates down the routing tree the corresponding time intervals, as follows. Each sensor receives an interval  $T_i = [start_i, end_i]$  from which it selects its own transmit times (time to sense, receive, compute and send) and further sends smaller sub-intervals  $T_{i,child_j} = [start_{i,j}, end_{i,j}] \subset T_i$  to its children. Thus, sensors learn their schedule and are properly synchronized. Our algorithms function correctly with both single and multiple BSs.

---

**Algorithm 1 : ETDMA (Efficient TDMA)**


---

```

end = -1;
for  $i = 0$  to  $baseStationCount - 1$  do
  start = end + 1;
  end = start + BS[i].timeSubtree - 1;
  do_ETDMA(BS[i], start, end);
end for

do_ETDMA(sensor,  $start_i$ ,  $end_i$ )
{
  sensor.sendStart =  $end_i - TRANSMIT + 1$ ;
  sensor.sendEnd =  $end_i$ ;
  sensor.compute =  $end_i - TRANSMIT - COMPUTE + 1$ ;
  if  $sensor.totalChildren > 0$  then
    sensor.recvEnd =  $end_i - TRANSMIT - COMPUTE$ ;
     $start_{i,j} = 0$ ;
     $end_{i,j} = start_i - 1$ ;
    for  $j = 0$  to  $sensor.totalChildren - 1$  do
       $start_{i,j} = end_{i,j} + 1$ ; // previous end + 1
       $end_{i,j} = start_{i,j} + (sensor.child[j]).timeSubtree - 1$ ;
      ETDMA( $sensor.child[j]$ ,  $start_{i,j}$ ,  $end_{i,j}$ );
    end for
    sensor.sense = ( $sensor.child[0]$ ).compute;
    sensor.recvStart = sensor.sense + 1;
  else
    // leaf node
    sensor.sense =  $start_i$ ;
    sensor.recvStart = -1;
    sensor.recvEnd = -1;
  end if
  sensor.timeAwake = sensor.sendEnd - sensor.sense + 1;
}

```

---

ETDMA, presented as Algorithm 1, assumes that switching the sensor from the listening state to the sleeping state is too costly and therefore does not do so lightly. The schedule serializes the actions of the leaf nodes, so that each such node stays awake for the minimum period of

---

**Algorithm 2 : OTAG (Optimal TAG)**


---

```

end = -1;
send = -1;
for  $i = 0$  to  $baseStationCount - 1$  do
  start = send + 1;
  send = start + BS[i].timeSubtree - 1;
  end = send - COMPUTE - TRANSMIT;
  do_OTAG(BS[i], start, end, send);
end for

do_OTAG(sensor,  $start_i$ ,  $end_i$ ,  $send_i$ )
{
  sensor.sendStart =  $send_i - TRANSMIT + 1$ ;
  sensor.sendEnd =  $send_i$ ;
  sensor.timeAwake =  $SENSE + sensor.totalChildren * RECEIVE + COMPUTE + TRANSMIT$ ;
  if  $sensor.totalChildren == 0$  then
    // leaf node
    sensor.sense =  $start_i$ ;
    sensor.compute =  $start_i + SENSE$ ;
    sensor.recvStart = -1;
    sensor.recvEnd = -1;
  else
    sensor.compute =  $end_i + 1$ ;
    sensor.recvEnd =  $end_i$ ;
    sensor.recvStart = sensor.recvEnd -
      sensor.totalChildren * RECEIVE;
    sensor.sense =  $sensor.recvStart - SENSE$ ;
     $end_{i,j} = start_i$ ;
    for  $j = 0$  to  $sensor.totalChildren - 1$  do
       $start_{i,j} = end_{i,j}$ ;
       $end_{i,j} = start_{i,j} + (sensor.child[j]).timeSubtree - TRANSMIT$ ;
       $send_{i,j} = end_i - (sensor.totalChildren - j - 1) * TRANSMIT$ ;
      OTAG( $sensor.child[j]$ ,  $start_{i,j}$ ,  $end_{i,j}$ ,  $send_{i,j}$ );
    end for
  end if
}

```

---

time and there are no collisions. All the non-leaf nodes wake up whenever their first child is sending a message to them, at time  $end_{i,0} - TRANSMIT$ <sup>1</sup>. The intermediate nodes stay awake until their last child sends its result, at time  $end_{i,totalChildren-1}$ . They then compute new aggregate values (if necessary) and transmit their results.

In Section IV we also evaluated two variants of this protocol. ETDMA-Opt1 schedules the sense and compute tasks in parallel for all leaf nodes. However, under both ETDMA and ETDMA-Opt1 nodes stay in the receive mode from the time when the first child transmits its result until the last child finishes its transmission. ETDMA-Opt2 improves ETDMA-Opt1 by allowing the BSs to go to sleep between two consecutive transmissions from its children. Since the

<sup>1</sup>We use the following constants: SENSE for the time necessary to sense the environment, RECEIVE for the time necessary to receive the readings from children, COMPUTE for the time necessary to compute the new aggregate values and TRANSMIT for the time necessary to send results to the parent.

---

**Algorithm 3 : timeSubtree(sensor)**

---

```
if sensor.totalChildren == 0 then
  // leaf node
  sensor.timeSubtree = SENSE + COMPUTE + TRANSMIT;
  return sensor.timeSubtree;
end if
sensor.timeSubtree = COMPUTE + TRANSMIT;
for i = 0 to sensor.totalChildren - 1 do
  sensor.timeSubtree += timeSubtree(sensor.child[i]);
end for
return sensor.timeSubtree;
```

---

time to transmit for each level  $i + 1$  subtree is usually significant and it is known to the BSs, this allows the level  $i$  nodes to save energy (we consider the BSs located at level 0).

OTAG, presented as Algorithm II, assumes that the transition cost to and from sleeping state is negligible and allows sensors to switch between states as necessary. OTAG is similar in functionality with the idealized version of the TAG protocol without any collisions, which can also be called *Perfect TAG* or *Optimal TDMA*. In this protocol, each sensor stays awake the absolute minimum time necessary to sense and receive messages from its children, compute and send its result.

OTAG represents an absolute, ideal lower boundary that we use to compare the performance of different versions of ETDMA. This protocol cannot be deployed in reality as-is because it unrealistically assumes that the time and energy costs to switch between states are negligible. Instead, each node can compute the energy consumption for staying awake and for going to sleep between two consecutive transmission from its children. Under ETDMA, each sensor knows that its children are going to transmit their result at the end of the interval it has previously sent to them. Based on this information, the node can decide if it is more beneficial for it to go to sleep or not.

### III. EXAMPLE

In order to illustrate the algorithms proposed in Section II, we consider the following example. The sensors are uniformly distributed in a  $5 \times 5$  grid topology and there are four BSs placed in the middle of the borders of the grid. A possible configuration of the routing trees (a) to (d) is shown in Fig. 2.

Due to space limitations, we present only two of the proposed schedules (ETDMA-Opt1 and OTAG). The trees can be scheduled in any order. Fig. 3 and Fig. 4 show the schedule for the routing tree (d) in Fig. 2, the schedules for the rest of the trees being similar. Each time slot is labeled with one of four letters (or a combination of them) plus the sensor identification number. These letters represent sensor states. **S** stands for **Sense**, **C** stands for **Compute**, **R** means **Receive** and **T** means **Transmit**.

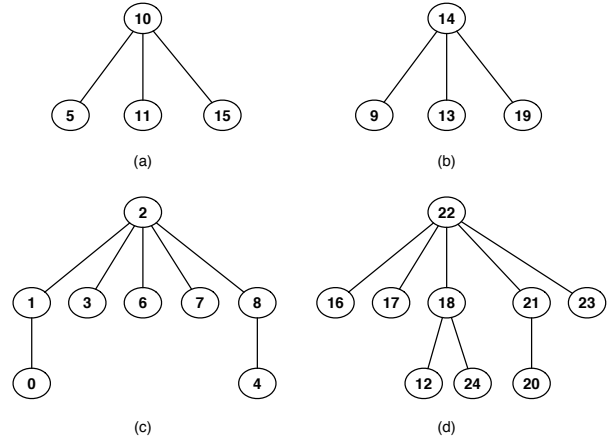


Fig. 2. Sensors Network Routing Trees

The length of the receive interval depends on the number of children the node has. To be more specific, it is a function of the number of descendants or nodes in its subtrees. The further the node is situated from the root, the sooner it can go to sleep and save energy. On the contrary, the closer the nodes are situated to the root, the more unbalanced the awake-time intervals become. This situation may also lead to very long receive intervals for some of the nodes, especially under ETDMA protocol or ETDMA-Opt1. OTAG represents the ideal protocol and nodes stay awake only when necessary to receive the transmissions from their children.

We have shown in [14] that multiple routing trees provide a lower number of levels on which sensors can find themselves in the hierarchy and also a more balanced number of nodes per level. All these ensure fairness for the upper nodes in the tree hierarchy (closer to the root), ensure long-living sensors network and finally increase the probability that more queries meet their deadline. The experimental results in this paper lead to the same conclusions. Even though our protocols are not dependent on a multiple routing trees architecture, they are able to benefit from it.

### IV. EXPERIMENTAL RESULTS

We created a simulation environment using CSIM [15]. We experimented with grid topologies with different sizes (from  $25 \times 25$  to  $45 \times 45$ ) and different numbers of BSs randomly placed in the network. We used the simulation parameters derived from [1]: sense and compute each take 1ms, receive and transmit each take 9ms and the collision backoff is 44ms. These values come from a 19.2 Kbps nominal rate and 20 byte message header, yielding  $8\text{ms} + 1\text{ms} = 9\text{ms}$  to transmit a message. According to [1], in the presence of collisions, the throughput drops to 4.43 Kbps, therefore the collision backoff is 44ms.

In this paper we present the most relevant findings, each result being reported after averaging the results from ten different runs. Different network configurations lead to different numbers of levels in the routing trees and different

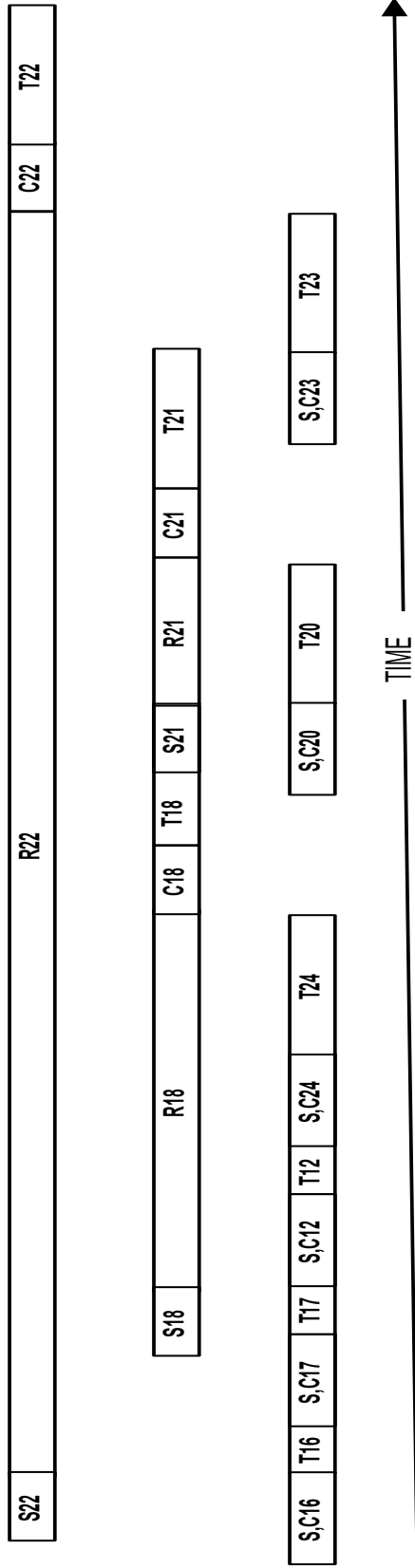


Fig. 3. ETDMA-Opt1 Protocol for subtree in Fig. 2(d)

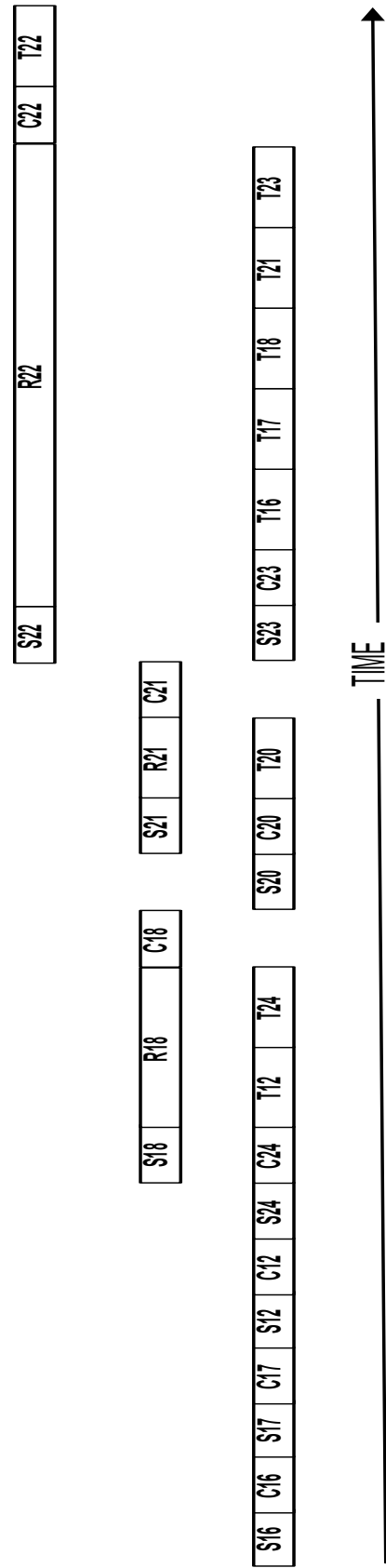


Fig. 4. OTAG Protocol for subtree in Fig. 2(d)

### 45\*45 Nodes, 1BS Grid

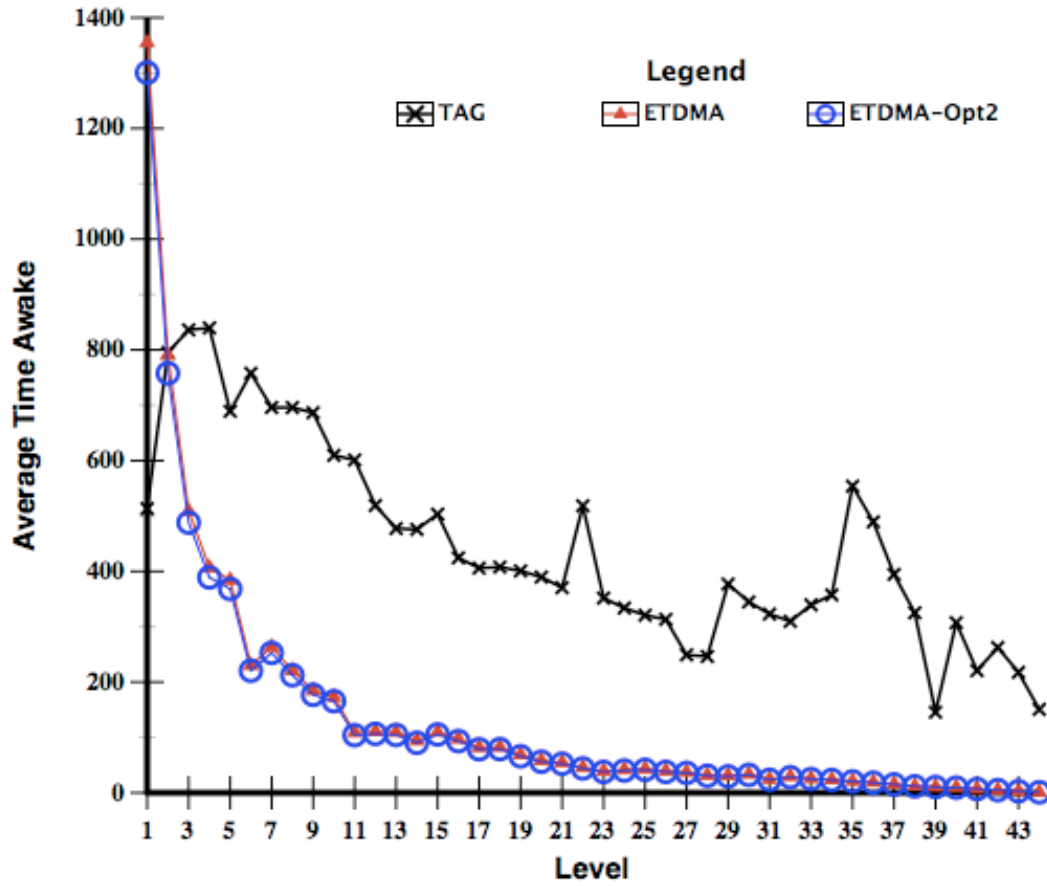


Fig. 5. ATAL as a function of time (1 BS)

### 45\*45 Nodes, 4BS Grid

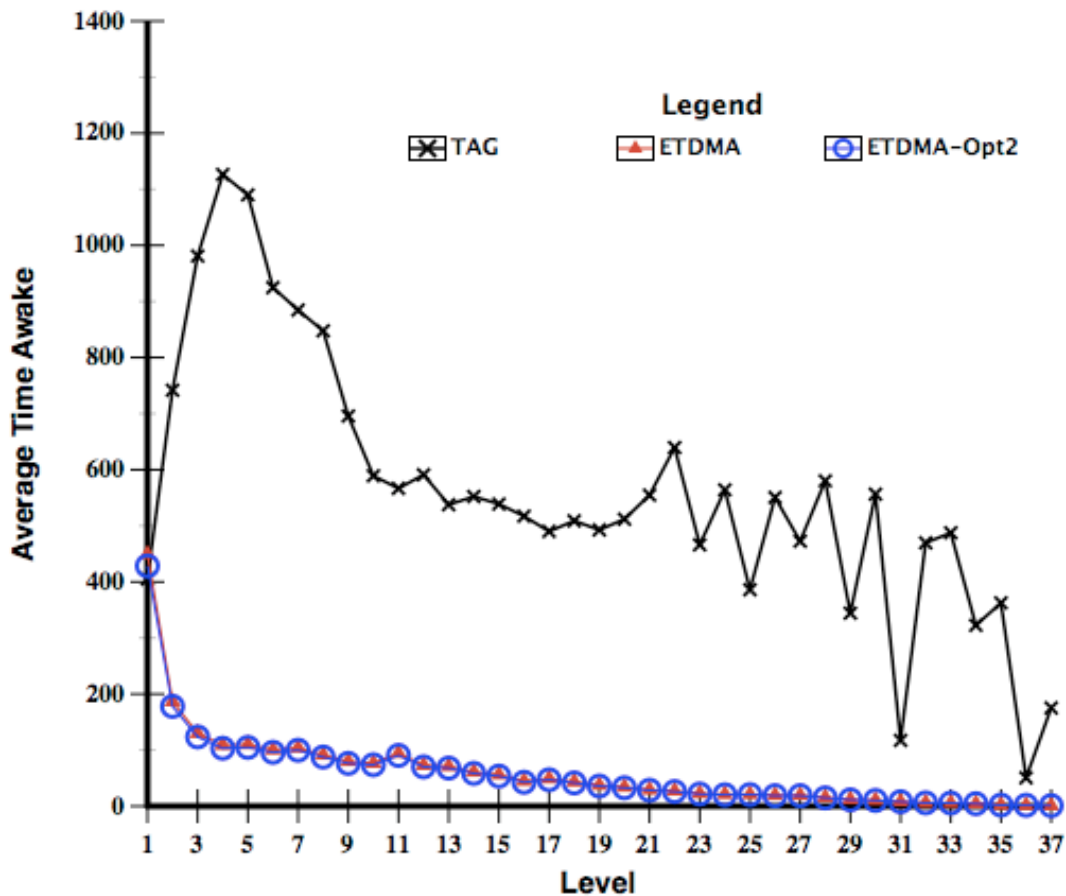


Fig. 6. ATAL as a function of time (4 BSs)

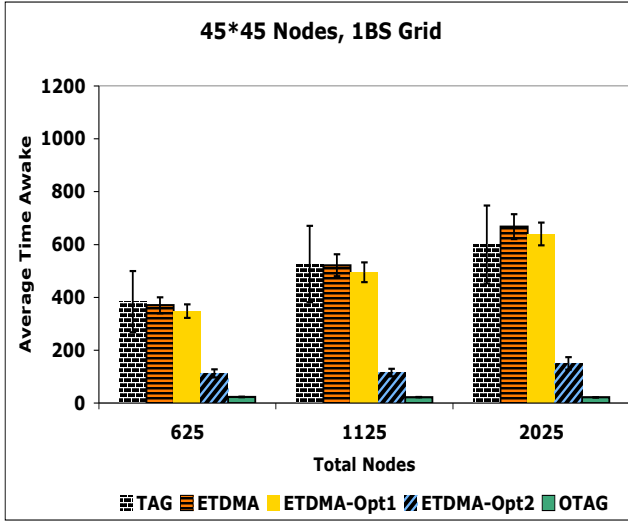


Fig. 7. ATA Distribution Summary (1 BS)

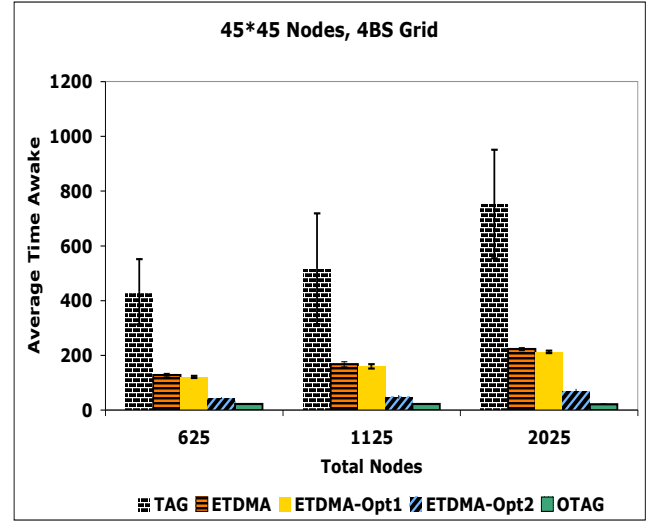


Fig. 8. ATA Distribution Summary (4 BSs)

distributions of nodes per level. We focused on analyzing the *Average Time Awake (ATA)* and *ATA per Level (ATAL)* for different network configurations.

We show the ATAL distribution in time for a 2025-node network configuration with one (Fig. 5) and four (Fig. 6) randomly placed BSs. For the root nodes, the ATAL values reported by TAG, ETDMA and ETDMA-Opt2 for one BS are 2739.1, 19507.1 and 80.3 and for four BSs are 1966.3, 4744.6 and 78.9, respectively. We can easily see that the ATAL value reported by ETDMA in this case is much bigger than any of the values reported by the rest of the protocols. In order to make these graphs easier to read, we decided to include a detailed view containing only the nodes situated on levels greater than zero. OTAG is not included in these figures, because its reported values are much lower than the rest of the protocols. It ranges from [1.1, 80.3] for one BS and [1.1, 78.9] for four BSs and it has an almost constant behavior, as sensors stay awake the minimum time necessary. ETDMA-Opt1 performs slightly better than ETDMA, as we can see in the figures containing the total ATA (Fig. 7 and Fig. 8). If we were to draw ETDMA-Opt1 in Fig. 5 and Fig. 6, it would follow very closely ETDMA. Note that both ETDMA and ETDMA-Opt1 benefit significantly from partitioning the sensor network in multiple routing trees; for example, the ATAL for level 0 decreases four times when we increase the number of trees from 1 to 4. ETDMA-Opt2 has a much lower ATA than ETDMA only on level 0 (BSs). For the rest of the levels, the two protocols have the same performance.

In Fig. 5 and Fig. 6 we can see the relatively smooth distribution of ATAL among levels for the proposed protocols. On the contrary, the TAG protocol has a very non-uniform distribution of ATAL. Our simulator did not capture all the time parallelism inherently found in TAG: each sensor must be awake from the moment its first child can possibly finish

its sensing and computing task (if it had the chance to run immediately) until the sensor itself transmits its result to its parent. If we were to employ the best collision detection for TAG (e.g. [9]), we could decrease the reported ATA to half. These new ATA values for TAG would still have been around two to three times larger on average than the values reported by ETDMA-Opt2 and around eight to ten times on average larger than the values reported by OTAG (depending on the number of the BSs in the network configuration).

We call the reader's attention to the particularly good results for the case of multiple routing trees. The improvement is more significant because the routing trees are shorter and more balanced, therefore spending minimum energy.

## V. DISCUSSION AND FUTURE WORK

There are three main potential problems to our proposed scheduling schemes.

*a) Failures and Mobility:* An issue with our schemes is how to react to failures in the network. Under our schemes, when a node fails (or runs out of energy), there is no way for the children to successfully transmit their values. The only option at this moment is to globally reconstruct the routing trees. Dealing with failures is a subject for further research.

Another potential problem is dealing with changing network topology. The case of sensors that disappear (move away) is the same case as having failures in the network. For sensors that are joining the network, one possible solution to this problem is to uniformly allocate some extra time slots. In this manner, selected nodes will listen an extra time slot to accommodate possible new children.

We have also identified some further optimizations for our scheduling schemes. In case some leaf children disappear from the network, their parents can decide to perform dynamic local reconstruction. The parent will shift the schedule for the remaining children and inform them about the changes. Periodically, the leaf nodes will stay awake for an

extra time slot to receive such messages. This optimization will make the schedule more adaptive and will save energy for the parent nodes. However, if many nodes disappear throughout the network, it is better to reconstruct the entire transmission schedule.

*b) Parallelism:* The TAG protocol, like many wireless MAC protocols, has inherent parallelism in transmissions, since it does not deal with reserved time slots. However, TDMA-like schemes can have only explicit parallelism.

Gaining back the parallelism in our schedules goes beyond ETDMA-Opt2. We can allow completely isolated routing trees to be scheduled simultaneously. The order in which routing trees are scheduled has a very big impact on the schedule's performance. Taking into consideration all the possibilities and checking for inter-dependencies among routing trees requires a complete search, which is far too expensive. We can apply the *A-Star* algorithm with different heuristics to prune the unnecessary branches from the possible solutions tree.

The ultimate goal would be to determine parts of the network belonging to the same or different routing trees that can be scheduled simultaneously. One possible solution is to carry out global graph coloring algorithms, which is very expensive. We can envision some approximation solutions. For example, we can globally require only information about the neighboring subtrees routed at a given level. This can be done in a top-down fashion (starting with level 1 and increasing the level until some parallelism is found) or in a random fashion (require information about a random level and if no success, repeat the process). However, more efficient heuristics are needed.

*c) Clock Synchronization:* This issue affects all TDMA-based scheduling schemes and it is related to exact clock synchronization among sensors. In the literature, there are several solutions proposed for this problem ([10], [3]). In this paper we assume loose clock synchronization.

## VI. CONCLUSIONS

In this paper we propose two efficient scheduling protocols for sensor networks. These protocols minimize the time and energy consumption for both creating and executing the schedule. The process of creating the schedule is fairly simple and it only requires a very small amount of knowledge about the time necessary for any given subtree to transmit its result. The experimental results suggest a significant (2-3 fold) reduction in average time awake per node compared to more traditional routing protocols like TAG.

## ACKNOWLEDGMENTS

This material is based on work supported by NSF under grants ANI-0125704 and ANI-0325353.

We would also like to thank Jonathan Beaver and Mohamed Sharaf for providing the initial simulator.

## REFERENCES

- [1] G. Anastasi, M. Conti, A. Falchi, E. Gregori and A. Passarella, Performance Measurements of mote Sensor Networks, *The 7th ACM International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, 2004.
- [2] M. Arumugam and S. S. Kulkarni, Self-Stabilizing Deterministic TDMA for Sensor Networks, *The 5th European Dependable Computing Conference*, 2005.
- [3] H. Attiya, D. Hay and J. L. Welch, Optimal Clock Synchronization under Energy Constraints in Wireless Ad-Hoc Networks, *The 9th International Conference on Principles of Distributed Systems*, 2005.
- [4] P. Bonnet, C. Choi and D. Mossé, Distributed Query Planning In A Sensor Network, *DIMACS Workshop on Pervasive Networking*, 2001.
- [5] Q. Cao, T. Abdelzaher, T. He and J. Stankovic, Towards Optimal Sleep Scheduling in Sensor Network for Rare-Event Detection, *The 4th International Symposium on Information Processing in Sensor Networks*, 2005.
- [6] U. Cetintemel, A. Finders and Y. Sun, Power-Efficient Data Dissemination in Wireless Sensor Networks, *The 3rd International ACM Workshop on Data Engineering for Wireless and Mobile Access*, 2003.
- [7] A. Demers, J. Gehrke, R. Rajaraman, N. Trigoni and Yong Yao, Energy-Efficient Data Management For Sensor Networks: A Work-In-Progress Report, *The 2nd IEEE Upstate New York Workshop on Sensor Networks*, 2003.
- [8] S. C. Ergen and P. Varaiya, TDMA Scheduling Algorithms for Sensor Networks, *Berkeley University*, 2005.
- [9] S. Gabriel, R. Melhem and D. Mossé, BLAM: an Energy-Efficient MAC Layer Enhancement for Wireless Adhoc Networks, *IEEE Wireless Communications and Networking Conference*, 2005.
- [10] T. Herman, NestArch: Prototype time synchronization service, *University of Iowa*, 2003.
- [11] T. Herman and S. Tixeuil, A Distributed TDMA Slot Assignment Algorithm for Wireless Sensor Networks, *The 1st International Workshop on Algorithmic Aspects of Wireless Sensor Networks*, 2004.
- [12] B. Hohlt, L. Doherty and E. Brewer, Flexible Power Scheduling for Sensor Networks, *The 3rd International Symposium on Information Processing in Sensor Networks*, 2004.
- [13] S. Madden, M. J. Franklin, J. M. Hellerstein and W. Hong, TAG: A Tiny Aggregation Service for Ad-Hoc Sensor Networks, *The 5th Annual Symposium on Operating Systems Design and Implementation*, 2002.
- [14] A. Munteanu, J. Beaver, A. Labrinidis and P. K. Chrysanthis, Multiple Query Routing Trees in Sensor Networks, *The IASTED International Conference on Databases and Applications*, 2005.
- [15] H. Schwetman, CSIM Users Guide, *MCC Corporation*, 1992.
- [16] D. Sharma, V. I. Zadorozhny and P. K. Chrysanthis, Structural Health Monitoring With Whirlpool, *The 5th International Workshop on Structural Health Monitoring*, 2005.
- [17] A. Sridharan and B. Krishnamachari, Max-Min Fair Collision-Free Scheduling for Wireless Sensor Networks, *The 23rd IEEE International Performance, Computing, and Communications Conference*, 2004.
- [18] N. Trigoni, Y. Yao, A. Demers, J. Gehrke, and R. Rajaraman, WaveScheduling: Energy-Efficient Data Dissemination for Sensor Networks, *The International Workshop on Data Management for Sensor Networks*, 2004.
- [19] Y. Yao and J. Gehrke, The Cougar Approach to In-Network Query Processing in Sensor Networks, *ACM Sigmod Record*, 31(3), 2002, 9-18.
- [20] V. Zadorozhny, P. K. Chrysanthis, P. Krishnamurthi, A Framework for Extending the Synergy between Query Optimization and MAC Layer in Sensor Networks, *The International Workshop on Data Management for Sensor Networks*, 2004.