

Cross-Layer Scheduling for Power Efficiency in Wireless Sensor Networks

Mihail L. Sichitiu

Department of Electrical and Computer Engineering
North Carolina State University
Raleigh, NC 27695-7911
mlsichit@ncsu.edu

Abstract—Wireless sensor networks are considered the sensing technology of the future. Large numbers of untethered sensor nodes can be used for tracking small animals and targets, environmental monitoring, enforcing security perimeters, etc. A major problem for many sensor network applications is determining the most efficient way of conserving the energy of the power source. Some networks use batteries, while others suggest different methods of gathering energy (e.g., solar cells). Regardless of the powering method, energy conservation is of prime importance for sensor networks. The best way to conserve energy is to turn the sensor nodes off; however, since an inactive sensor node is no longer part of the network, the network can become disconnected. This creates a fundamental trade-off. In this paper, we propose a deterministic, schedule-based energy conservation scheme. In the proposed approach, time-synchronized sensors form on-off schedules that enable the sensors to be awake only when necessary. The schedule establishment is fully distributed and thus appropriate for large sensor networks. The performance of the proposed approach is evaluated through the use of simulations.

Keywords: System design, Simulations.

I. INTRODUCTION

Wireless sensor networks [1]–[3] have the potential to revolutionize sensing (and actuating) technology in the future. Large numbers of cheap untethered nodes can be placed in the area to be monitored. The large number ensures that at least some of the sensors will be close to the phenomenon of interest and thus be able to have high quality measurements. In-network processing allows tracking of targets and the evolution of the studied phenomena. It also allows for substantial power savings and reduced bandwidth necessary to observe certain phenomena [4]–[10]. The large number of sensors also increases the reliability of the system, as failure of a percentage of the sensor nodes will not result in system failure.

Like any other electronic devices, sensor nodes have to be powered. If a power cable is used, many of the advantages enabled by the wireless communications are voided. In most applications, a power cable is not an option. A very popular method of powering wireless sensor networks is with the use of batteries. An alternative is to harvest energy, for example, from a piezoelectric element or from solar panels. However, under any reasonable assumption, onboard power will be limited and must be expended responsibly.

Sensor nodes are expected to have very small form factors. They are also expected to be inexpensive and deployed in very large numbers. This precludes spending a large amount of resources on a large, expensive power source for each node. Once deployed, the sensor networks are usually unattended; therefore battery replacement is out of the question – the life of the sensor network is determined by the life of its batteries. Finally, such a network is typically expected to work for extended periods of time (weeks, months, and in some cases, years).

There are several sources of power consumption in sensor networks, and correspondingly different methods of reducing power consumption [11]:

- **Idle listening** is the major power consumption source for many networks. For most transceivers, the receive mode power consumption is on the same order of magnitude as the transmission power [12]–[14], and most MAC protocols put the transceiver in receive mode whenever it does not transmit, whether there is the need to receive a message or not.
- **Retransmissions resulting from collisions** can be quite significant if the network load is high and the collisions frequent.
- **Control packet overhead** (e.g., RTS, CTS, ACK) can be significant for sensor networks which, typically, have small packets.
- **Unnecessarily high transmitting power** not only results in higher power consumption, but may also increase the interference at other nodes in the network.
- **Sub-optimal utilization of the available resources**; for example, routes that utilize the nodes with the largest (remaining) batteries should be preferred.

Corresponding to the importance of the problem, there is a significant body of research addressing different aspects of the power control problem [4]–[36]

Some approaches limit the transmission power aiming to increase the spatial reuse while maintaining network connectivity [15]–[17]. Especially for ad hoc networks, but also relevant for sensor networks, power aware routing protocols [25]–[30] demonstrate significant power savings. Approaches at the MAC layer [11]–[13], [18]–[24] turn the wireless transceivers off whenever they are clearly not needed (e.g.,

during backoff periods). The main power savings in these papers results from reducing the idle listening power, but also from decreasing the number of collisions.

Depending on the specific applications (or classes of application), several approaches at the application layer [4]–[10] may dramatically improve power consumption. It becomes increasingly clear that power efficiency design cannot be addressed completely at any single layer in the networking stack [14], [36].

For wireless local area networks (WLANs), several power saving approaches have been standardized for IEEE 802.11 [37] and Bluetooth. In WLANs, the problem is significantly simpler than in ad hoc networks due to the existence of a single coordination point (access point for 802.11 and the master node for Bluetooth).

We will differentiate between *event driven* and *continuous monitoring* sensor networks. A detailed explanation of the two modes can be found in [10]. The differentiation is mainly dictated by the application. In an event driven sensor network, the sensor nodes do not send data (and are most likely asleep) until a certain event occurs. For example, in a forest fire monitoring application, until smoke or fire is detected, no data needs to be sent. The main difficulty in an event driven sensor network is to be able to wake up the entire network (or at least a path to the base station) when the event occurs. In a continuous monitoring sensor network, data is sampled and transmitted at regular intervals. For example, a temperature monitoring station can take a reading every half hour and send it to a central monitoring station. Implementation details and compression schemes tend to blur the distinction between the two classes of sensor networks. Indeed, many times, in order to detect an event, a continuous monitoring scheme is necessary (e.g., query the smoke sensor once a minute). Similarly, if only temperatures different from the previous reading are sent, the temperature change can be defined as an event. Both schemes can be easily implemented using an event driven operating system (e.g., TinyOS [38]).

Many existing power saving approaches address the needs of general ad hoc networks where mobility and unpredictable traffic patterns impose a trade-off between network performance (delay and throughput) and power savings. In contrast, this paper presents an approach tailored specifically to the needs of sensor networks with continuous monitoring capabilities. The proposed scheme derives its power efficiency from eliminating (almost completely) idle listening and collisions in the sensor network. The need for such a scheme is highlighted in [10] and prompted by recent habitat monitoring applications [39], [40].

II. PROPOSED APPROACH

There is no better way to conserve energy than to put the nodes to sleep (since using low power components only goes so far). However, a node that is sleeping is no longer part of the network, and thus cannot help to deliver the sensor data from its neighbors to its destination. This creates a fundamental trade-off.

Taking advantage of the unique characteristics of sensor networks (like stationarity and long-lived, predictable data flows), we propose to develop a framework for deterministic optimal energy conservation while maintaining the network real-time characteristics. The idea is simple, yet powerful. In the proposed approach, sensor nodes dynamically create on-off schedules in such a way that the nodes will be awake *only when needed* and asleep the rest of the time.

The proposed scheme can be decoupled into two distinct phases *for each flow* in the network:

- **The setup and reconfiguration phase** takes place during the initialization of the network, and subsequent to any changes in the network queries and the availability of the routes. The setup and reconfiguration phase is relatively short in comparison to the steady state phase. Its goal is to set up the schedules that will be used during the steady state phase. If compared to the routing and forwarding engine of an IP network, the setup phase corresponds to writing the routing tables.
- **The steady state phase** takes place between consecutive setup and reconfiguration phases. Similar to the forwarding phase in a routing engine, the steady state phase is the workhorse of our scheme: it utilizes the schedule established in the setup and reconfiguration phase to forward the data to the base station.

These two modes of operation call for two very different ways of managing communication. During the setup and reorganization phase, the network needs to self-organize in a distributed fashion in order to achieve its goals, while maintaining power efficiency and robustness. During the steady state phase, the network must operate using as *regular* a schedule as possible, to favor maximum efficiency.

We assume that the traffic is periodic, with the same period in the entire network. Furthermore, we assume that each node originates only one packet in each period (also called an epoch in [10]). This corresponds to one data flow. If multiple packets are sent in each period, a setup phase should be initiated for each of these packets, or they can be grouped together in a single data flow. A data flow is at a given time either in the setup and reconfiguration phase, or in the steady state phase. Different flows in the network can be in any of the two states at a given moment. Therefore, new nodes (and the associated flows) can join and leave the network without disturbing the rest of the flows. To ensure that the control packets necessary to set up schedules in the setup and reconfiguration phases do not collide with the data packets forwarded in the steady state phase, a two-level priority scheme MAC layer has to be used (we will provide further details in Section II-B).

Figure 1 depicts the two phases and three states corresponding to the flow of each active node in the network. The network topology can change because of node failures, battery depletion, or external influences. All flows affected by such a topology change will have to enter the setup and reconfiguration phase until a new route is established. We expect that the time scale of the topology changes (e.g., a change every day) and of the data forwarding (e.g., a packet

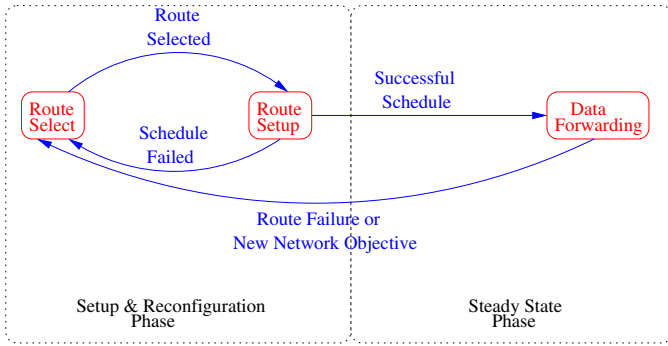


Fig. 1. State diagram for each data flow in the network.

every second) to be significantly different; therefore, every flow will spend a large percentage of its time in steady state phase when useful work is performed and very little time in the setup and reconfiguration phase, which is pure overhead, as far as data forwarding is concerned.

Another distinct reason for entering the reconfiguration phase is a change in the network objective (corresponding to a different query [10] or query parameters).

A. The Steady State Phase

Assume for the moment that the network is perfectly synchronized. We will shortly relax this rather unrealistic assumption, and we will study the effect of less than perfect synchronization on the efficiency of the scheme in the simulation section (Section III).

In the steady state phase of a flow, the nodes on the path simply forward the DATA packets originated by the source of the flow at the appropriate times. To this end, each node on the path stores a *schedule table* that specifies *when* various *actions* have to take place. The three different actions considered in this paper are:

- **Sample**, corresponding to the source node taking a data sample. This sample will be forwarded along the (generally multihop) path to the base station. Samples from multiple queries can be grouped together and considered part of the same data flow, or sent as different data flows. In any flow only the source of the flow has a sample action in its schedule table.
- **Transmit**, corresponding to the action of transmitting a packet of the flow to the next node on the path to the base station. All nodes on the path of a flow, except for the base station, have a transmit action associated with the flow.
- **Receive**, corresponding to the reception of a packet. This packet will be further transmitted to the next node in the path until it reaches the base station. All nodes on the path of the flow, except for the source node, have a receive action associated with the flow.

Figure 2 depicts the actions associated with one flow originating at the source node $S = i_0$, being forwarded through the intermediate nodes i_j , $j = 1, \dots, M$ to the base station $BS = i_{M+1}$. Each node on the path of i_j , $j = 0, \dots, M + 1$ has

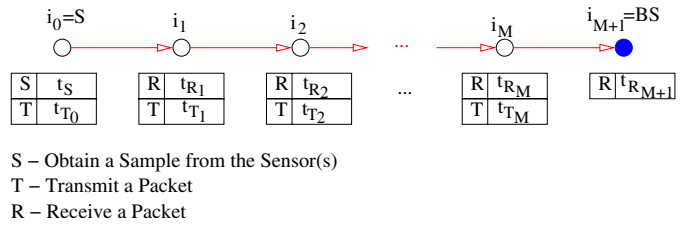


Fig. 2. The actions associated with a flow originating at the source S and being forwarded to the base station BS.

two actions associated with this particular flow. The actions are stored in a schedule table that has two columns:

- the first column stores *what* type of action that has to be performed; and
- the second column stores *when* a certain action has to take place (i.e., a time); since, for the moment, we assume the network is perfectly synchronized, time is consistent throughout the network.

Assume that Δ_S seconds are necessary to obtain a sample. Assume that the transceiver has to be Δ_R seconds in receive mode to be able to receive a packet, and Δ_T seconds in transmit mode to be able to transmit a packet. If the network is perfectly synchronized, $\Delta_T = \Delta_R$. With these assumptions, causality constrains the times in the schedules as follows:

$$t_{T_0} \geq t_S + \Delta_S, \quad (1)$$

$$t_{T_j} \geq t_{R_j} + \Delta_R, \quad \forall j = 1, \dots, M. \quad (2)$$

If the clocks are perfectly synchronized, the receiver of node $j + 1$ has to be started at the same time as the transmitter of node j :

$$t_{R_{j+1}} = t_{T_j}, \quad \forall j = 0, \dots, M. \quad (3)$$

Since perfect synchronization is practically impossible, an existing synchronization approach [41]–[44] can be used to synchronize adjacent nodes with a finite precision. We will study the effect of the finite synchronization precision in Section III. To ensure that the receiver is enabled by the time the transmitter starts the transmission, guard times are needed. Thus, if nodes j and $j + 1$ are synchronized with a precision of $\pm\Delta$, (3) becomes:

$$t_{R_{j+1}} = t_{T_j} - \Delta, \quad \forall j = 0, \dots, M, \quad (4)$$

and the awake time of the receiver has to be increased by the same amount:

$$\Delta_R = \Delta_T + \Delta. \quad (5)$$

In addition to conditions (1)-(4), any two transmissions that can interfere with each other have to be scheduled at non-overlapping times. We will show in Section II-B how such a schedule can be formed in a distributed manner.

To illustrate network operation in the steady state phase, consider the situation depicted in Fig. 3 with 10 sensor nodes (1-10) connected to one base station (BS) that may further relay the data back to a monitoring station via a long-range radio. We depict the active routes used by the wireless sensor

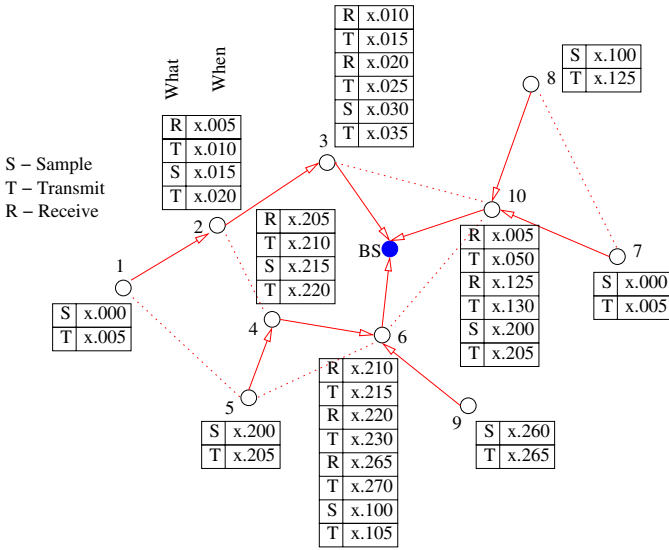


Fig. 3. Example of a 10-node sensor network with a base station BS and a possible distributed schedule.

nodes by continuous lines with arrows and alternate (backup) routes by dotted lines. The proposed scheme works with variable-length packets; but for simplicity, assume that each sensor has to make a measurement once a second and that it takes 5 ms to take a sample, and another 5 ms to send the data.

The leaf nodes (e.g., node 1) do not have to be awake for more than 10 ms every second. They need to wake up, take a sample (5 ms) and send it (5 ms), then go to sleep for 990 ms. For example, node 1 can wake up at times instants 1.000, 2.000, 3.000, etc. for 10 ms and sleep the rest of the time. In the schedule table of node 1, the fact that it has to wake up at 1.000, 2.000, 3.000,... to take a sample is denoted by an “S” and x.000.

Intermediate nodes (e.g., node 2) on the other hand, have to take measurements *and* relay messages on behalf of other nodes. For example, node 2 needs to be awake for 10 ms to forward the message from node 1 (receive for 5 ms, then forward for 5 ms), then take its measurement and forward it (another 10 ms). In all, it has to be awake for 20 ms, then go back to sleep for 980 ms. In the schedule table of node 2, the receiving action is marked by an “R”, and the transmitting action is marked by a “T”.

In the proposed system, each node will form and maintain scheduling tables similar to the ones shown in Fig. 3. The scheduling table of a node contains two entries for all of the flows originating or forwarded by that node (S and T entries, if the node originates the flow, and R and T entries, if the node forwards the flow).

Comments:

- If the base station has limited energy as well, it can also maintain a schedule consisting of “R” (receive) entries.
- The approach does not limit the capacity of the network. If data forwarding cannot be scheduled, then the capacity

of the network is exceeded. In fact, it can be shown that the capacity of the network is maximized when a synchronized schedule like the one presented in this paper is implemented [45].

- The nodes closer to the base station will have longer schedules, as they are required to forward data on behalf of the nodes closer to the periphery of the network. This undesirable effect is inevitable in a multihop architecture with base stations.
- Notice that we did not discuss what happens when a transmission fails. There are several options:
 - If the application tolerates it, we can simply ignore lost packets.
 - In case the medium access control (MAC) layer supports retransmissions, enough time can be reserved in the schedule for several (but a limited number of) retries in case the packet does not go through the first time.
 - Alternatively, the schedule can have one (or several) special spare cycle(s) reserved specifically for retransmissions.

If retransmissions are used, we expect the lifetime of the network to decrease roughly with $1 - \rho$ where ρ is the packet error rate. (The expected number of transmissions for each packet is $\frac{1}{1-\rho}$.) For small values of ρ , the effect is negligible.

- We expect the energy savings to decrease with the decrease in the precision of the synchronization. Indeed, according to (5), Δ , the synchronization precision, is equal to the idle listening time.
- Even ignoring the power wasted due to guard times, the fact that a node sleeps $x\%$ of the time does not imply that $x\%$ of the power will be saved. As it is correctly pointed out in [14], the power spent to wake up is not negligible. An important side-effect of the wake-up overhead is that “compact” schedule (like the one for node 3 in Fig. 3) is preferable to a schedule with gaps (like the one for node 6), which forces a node to wake up several times during one time period.
- To provide robustness, in case an exception occurs in the steady state phase (e.g., a collision occurs, several consecutive packets are lost, the synchronization algorithm fails and has to be restarted, etc.), the affected flows will reenter the setup and reconfiguration phase that will attempt to repair the exception.
- For a given network topology and traffic parameters, there may be many schedules that satisfy the required non-interference conditions. As far as power savings are concerned, they are all equivalent: each node except for the base station will have to be awake for two actions for each flow it handles. Therefore, there is no question of optimality from the power savings point of view; however, different criteria can be used to differentiate two schedules:
 - **Compactness:** As mentioned before, a compact

– **Compactness:** As mentioned before, a compact

schedule at each node is preferable, as the wake-up power overhead will only be expended once per period.

- **Delay:** A schedule that minimizes the total delay in the network (or alternatively, the maximum delay) may be desirable for certain applications.
- **Load balancing:** A schedule that equally distributes the burden of forwarding among nodes may be able to increase the time to the first node death due to battery starvation in the network.

The advantages of this scheduled approach are multiple:

- First and foremost, it enables power savings by completely (almost completely with realistic synchronization) eliminating idle listening, as nodes can go to sleep whenever they do not *have* to be awake.
- A proper schedule avoids packet collisions almost completely. (Rare collision with control packets are still possible as we will show in Section II-B.) This may increase the capacity of the sensor network several times and simplifies the design of the MAC layer.
- A side-effect of avoiding the collisions is that each packet will experience a small delay and practically no delay jitter, as randomness is practically eliminated from the network (barring transmission errors which require retransmissions).
- Very little or no buffering is required at intermediate nodes, as each is guaranteed to be able to forward the data before other packets are received.

B. The Setup and Reconfiguration Phase

Of course the interesting question is, “How is the schedule presented in the previous section discovered and maintained in an ad-hoc and distributed fashion?”

As suggested in Fig. 1, the schedule setup algorithm for any flow proceeds in two steps:

- In the first step (*route select*) a route from the node originating the flow to the base station is selected. It is the role of the routing protocol to provide this route.
- In the second step (*route setup*) the schedules are set up along the chosen route. To this end, a special route-setup packet is sent along the route from the source node of the flow to the base station. The route setup packet is building the schedule at the intermediate nodes as it travels toward the base station.

1) *The Route Select Step:* The setup and reconfiguration algorithm is *independent* of the underlying routing algorithm. Therefore, many of the algorithms available for routing in ad hoc and sensor networks [10], [13], [46]–[51] can be used. Power aware routing algorithms [25], [26] may be preferable, as they have been shown to provide substantial increases in network lifetime. The only requirement for the routing algorithm is to provide at least one route from each node to the base station.

We will show how the setup phase proceeds using a particular routing protocol. Let us consider a routing protocol where

the base station advertises routes using a distance vector protocol. The advertisements can be part of a query disseminated in the network. The advertisements are periodically flooded in the sensor network. Each node will thus be able to find how many hops it is from the base station. Subsequently, each node chooses one parent with a smaller hop count than itself as its default route. Therefore, the network will self-organize in a routing tree with the sink at the base station [10]. If multiple choices of the next hop exist, then a parent can be selected using link quality [13], application layer considerations [10], or any other metric that attempts to minimize delay, maximize network lifetime [25], [26], etc.

2) *The Route Setup Step:* During the route setup step, a special route setup (RSETUP) packet will be sent on the selected route from the source of the flow to the base station. At each intermediate node, the RSETUP packet has two distinct goals:

- to find a time when a DATA packet can be scheduled without colliding with other nearby transmissions, and
- to append the appropriate entries in the schedules of the two nodes.

At the minimum, the RSETUP packet will contain the node source and the duration of the packets on that flow (Δ_T).

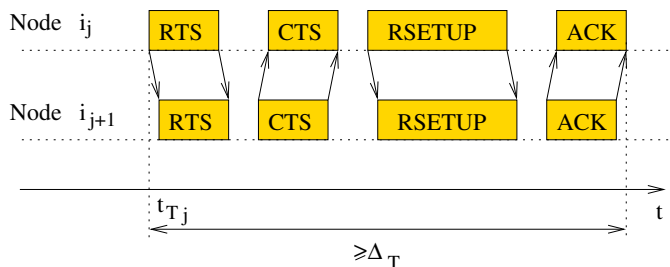


Fig. 4. An RTS/CTS exchange of an RSETUP packet between nodes j and $j + 1$.

To ensure that only non-contending transmissions are scheduled at the same time, the MAC protocol of the RSETUP packet needs to protect both ends of the communication link. An example of such a protocol is the well known RTS/CTS family of protocols [37], [52], [53] (see Fig. 4). Since it is shown that, under certain circumstances, a collision can remain undetected by an RTS/CTS type of protocol [54], a solution like the one presented in [54] can be employed. With a properly configured RTS/CTS MAC protocol [54], it can be ensured that contending transmissions are not scheduled at the same time.

At each intermediate link an RSETUP packet will set up a pair of actions at the two nodes that form the link: a transmit action for the node i_j , and a receive action at node i_{j+1} (see Fig. 5). The transmit time t_{T_j} is saved at the time the transmission of the RTS starts. As Fig. 4 suggests, to avoid concurrent transmissions when DATA packets will be sent, the duration of the setup transmission has to be greater or equal to the duration of the transmission of a DATA packet for that flow Δ_T . The time $t_{R_{j+1}}$ is computed using (4).

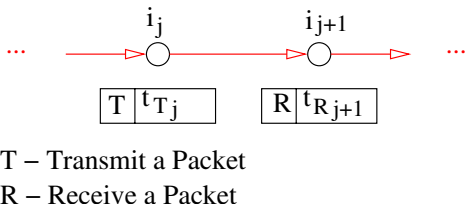


Fig. 5. Upon a successful exchange between nodes j and $j + 1$, a pair of actions is appended to the schedule tables of the two nodes.

If the transmission of the RSETUP packet is successful (i.e., acknowledged), both node i_j and i_{j+1} enter the corresponding actions in a temporary schedule. The actions in the temporary schedule will be copied in the permanent schedule when the route acknowledgment is sent from the base station on the reverse path, or are purged upon timeout. The entries in the permanent schedule are also purged if a number of consecutive DATA packets are missing (a simple counter scheme can be used in the implementation).

If the transmission of the RSETUP packet is not successful (e.g., the carrier is sensed at time t_{T_j} , no CTS or no ACK is received), transmission is postponed at the earliest time when a packet is not to be sent or received at node i_j . If the packet is postponed for more than a period, a route error (RERR) is sent back to the source. To allow the error message to reach the source, the route should be recorded in the RSETUP packet. Alternatively, for each flow, each intermediate node can store both the next hop and the previous hop information.

If the RSETUP packet arrives at the base station, a route acknowledgment (RACK) packet will be sent from the base station on the reverse path to the source. Each node on the reverse path will move the scheduled entries from the temporary schedule to the permanent schedule upon the receipt of the RACK packet.

It is very important that the control packets (RSETUP, RACK and RERR) do not disturb the existing data flows. Therefore, the control packets will have a lower priority than DATA packets sent on already scheduled flows. We propose a simple two priority schedule system, similar to the 802.11 [37] priority system:

- DATA packets are sent immediately when scheduled, without performing a carrier sense. Optionally, if reliability is important, an ACK can follow the transmission of the data packet.
- The control packets are sent after waiting for a small delay (similar to SIFS in 802.11) and after sensing the carrier. The MAC layer of the control packets is based on RTS/CTS, as we previously discussed.

This simple priority scheme avoids most collisions (but not all). Since conflicting DATA packet transmissions are scheduled at different times, carrier sensing is not necessary; moreover, not sensing the carrier for DATA packets is essential for the correctness of the scheme: if an RSETUP packet handshake partially overlaps a DATA transmission, the DATA transmission will have to be sent (even if it will be

compromised) to ensure that the RSETUP exchange is not successful. This way, an RSETUP packet will be successful if, and only if, it does not overlap (even partially) with a DATA packet.

Therefore, collisions may occur during the setup phase of the flows between data packets and control packets; however, since reconfiguration seldom happens, those collisions will have a very small influence on the efficiency of the scheme.

After receiving a RACK packet from the base station, the source enters the steady state phase. If a RERR packet is received, alternate routes can be explored until a successful route is established or all alternate routes are exhausted. If no alternate route can be established, the capacity of the network is exceeded; i.e., the particular flow cannot be carried by the network. The bound on the number of flows that can be scheduled is not given by the scheduling approach, but rather by the capacity of the network.

Of course, for node i_{j+1} to be able to receive the RSETUP packet from node i_j , it has to be awake when the RTS request comes from node i_j . Therefore, to allow the setup of new routes, it is imperative that all nodes, in addition to their scheduled time, will be awake an additional short time just for setting up new schedules. The setup slot can be at a fixed time throughout the network (e.g., at x.000); or, to favor compact schedules, it can be immediately after the last event in the schedule of the node. We expect this extra setup time slot to be the largest overhead of the scheme, as this slot will go unused in almost every time period (assuming that reconfiguration happens infrequently). Adaptive schemes trading flow setup time for power savings can be employed to save even this extra setup slot (e.g., by keeping the node awake in only one out of ten of those extra slots).

The pseudocode of the proposed scheduling algorithm is presented in the Appendix.

Comments:

- The route setup phase, while being a typical layer 3 function, in the proposed approach relies heavily on layer 2 information and properties. Moreover, different MAC layers for the control and data packets are essential to the correct functioning of the scheme.
- The schedule can take full advantage of the spatial reuse inherent in ad-hoc networks. For example, in Fig. 3, nodes 1 and 7 are scheduled to send at the same time, as their transmissions do not interfere. The scheduling algorithm ensures that only transmission that can be simultaneously scheduled may be scheduled simultaneously.
- Schedule tables should be able to maintain “per flow” information (two entries for every flow). This should not be a big problem for moderately large networks; however, if it becomes a problem, the requests and table entries can be aggregated (e.g., between time x.025 and x.130 forward any received packets).
- A different routing protocol can be used for the first step of the route setup scheme. For example, an on-demand

routing protocol can be employed. The source node trying to reach the base station will collect (all) route replies before it will send the RSETUP packets toward the base station on the discovered routes.

- If in-network processing/aggregation [4]–[10] is used, the same route setup procedure can be used while replacing the base station with the node that performs the aggregation.

III. SIMULATION RESULTS

To evaluate the performance of the proposed approach, we compared the energy savings of the proposed scheme with those of a sensor network with an 802.11-like power saving mode (PSM) and with a sensor network without any energy savings features (i.e., with nodes in receive mode whenever they do not transmit), but otherwise identical parameters.

It became very quickly apparent that current ad hoc network simulators (ns-2, Glomosim, Qualnet, Omnet++, OPNET, etc.) cannot simulate networks of hundreds of nodes for periods of months, or years. The main problem encountered with these simulators is their very detailed modeling of the lower layers (physical and MAC), which consumes lots of time. We chose to implement our own simulator that idealizes the physical and MAC layers.

The nodes are assumed to be powered by two AA alkaline batteries with a capacity of 2000 mAh. The battery self-discharge [55] phenomenon was not modeled. Since the focus of the paper is not on the routing algorithm, a simple shortest path algorithm was employed. If multiple shortest paths with different amounts of power left are available, the one with the largest power available is preferred. We assumed that no in-network processing is done, and therefore, all data is forwarded (in a multihop fashion) to the base station. We assumed that the network has enough capacity to carry all packets, and that no collisions occur in either of the three types of networks simulated. We assume that the base station has enough power to outlast any of the sensor nodes.

An interesting question is, “What is the cost function to be compared?” The average and maximum power consumed in the network are two alternatives; however, the goal of the sensor network is not to conserve power, but rather to forward the sensed data. Therefore, the time until a node fails due to energy depletion should be a better cost function than the power consumed in the network. However, it is likely that even if a node fails, the network can be reconfigured such that the forwarding function of the failed node can be relegated to neighboring nodes. Sensor networks are designed redundantly, to be able to withstand the failure of a significant portion of their nodes; however, when most of the nodes have depleted their batteries, or are unable to reach the base station, the sensor node will be unable to fulfill its function. Therefore, we will use *the time until a certain percentage of nodes can no longer forward data to the base station (either because their batteries are depleted, or because there are no routes to the base station)* as the main performance measure. The exact percentage of nodes that can be lost without considering

a system failure depends on the particular application and degree of redundancy built into the sensor network. We assume that the network capacity is not reached, such that a feasible schedule exists.

Upon node failure, the network is reconfigured: the nodes that had flows that used the failed node will find the new shortest paths (if available) to the base station and will start to forward data as soon as such a path is scheduled. To separate network reconfigurations due to physical node failure and the ones due to battery depletion, we did not model physical node failure. Since the proposed approach does not guarantee compact schedules, we considered the worst case scenario, where a node has to go to sleep and wake up for each packet to be forwarded. An extra control and retransmission slot is added to the schedule of each node.

To decouple the effects of various parameters on the performance of the scheme, we started from a base case and varied one parameter at a time.

For the base case, we simulated $N = 100$ nodes and a single base station randomly deployed in a $100m \times 100m$ rectangular area. The transmission range is assumed circular with a radius of $R_{TX} = 25m$. We considered a system failure when $p = 50\%$ of the nodes were unable to forward their data to the base station (i.e., they either ran out of batteries or had no routes to the base station).

We assume that each sensor node sends a single packet every $T_P = 60s$. We assume that each packet takes $T_{TX} = 50ms$ to send, and that each node can transition from sleep mode to a fully awake state in $T_{WU} = 3ms$. We assume the precision of the synchronization to be $\Delta = 1ms$ (even better precision has been demonstrated [41], [42]). The beacon interval for the 802.11 power savings mode is $500ms$ (values consistent with the IEEE 802.11 defaults do not make sense in a sensor network setup that features significantly slower data rates than IEEE 802.11).

We used power consumption values similar to the ones available for the Berkeley motes [12], with the transmission current $I_{Tx} = 17mA$, the receive current $I_{Rx} = 10mA$, the start-up current $I_{WU} = 5mA$ (active MCU, inactive transceiver) and the idle current $I_{idle} = 10\mu A$.

	Network Lifetime	
	Mean	Std. Deviation
No power savings	8.3 days	4 minutes
802.11 PSM	3.2 months	7.5 days
Power scheduling	24.2 months	5 months

TABLE I
AVERAGE NETWORK LIFETIMES FOR THE BASE CASE USING THE THREE POWER SAVING STRATEGIES.

The simulation results for the network with the parameters described above for the base case are presented in Table I. It is clear that the proposed approach can significantly increase the network lifespan. Since the standard deviation for the proposed schedule is significant (due to variability in the initial topology), we will present average results for all of

the following experiments.

In other words for the network with no power savings, the lifetime can be computed precisely, regardless of the initial node positions, while for the proposed power saving scheme, the lifetime of the network can vary significantly with the initial placement of the nodes. Therefore, in what follows, average values are computed for all measurements.

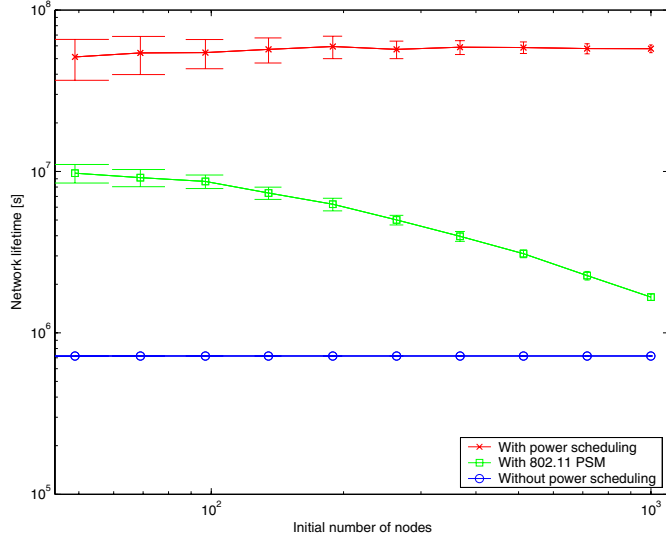


Fig. 6. Dependency of the network lifespans on the number of nodes for a constant deployment area.

Figure 6 depicts the dependency of network lifespan on the total number of nodes while keeping the deployment area constant ($100m \times 100m$). The lifespan of the network with the power schedule and of the one w/o power schedule are approximately constant with the increase in the number of node. This is due to the fact that adding new nodes in the same area does not increase the depth of the network, and thus the forwarding overhead remains constant with the number of nodes. The 802.11 PSM network performance decreases with the increase in the number of nodes, because now, at a beacon interval, the nodes will have to be awake longer (on the average) until all nodes have transmitted their packets.

In contrast, in Fig. 7, we kept the density of the network constant, while increasing the number of nodes (and, correspondingly, the deployment area). The forwarding overhead, in this case, increases with the number of nodes, and many nodes close to the base station have to be awake most of the time to forward packets from nodes outside the reception area of the base station. The nodes close to the base station tend to deplete their batteries first; after a relatively short time, the rest of the network cannot bridge the gap around the base station, resulting in premature system failure (most nodes still have plentiful energy supplies, but no routes to the base station).

A crucial parameter of the system is the measurement period (referred to as “epoch” in [10]) of the continuous monitoring sensor network (Fig. 8). Increasing the period effectively reduces the “duty cycle” of the network. A larger period

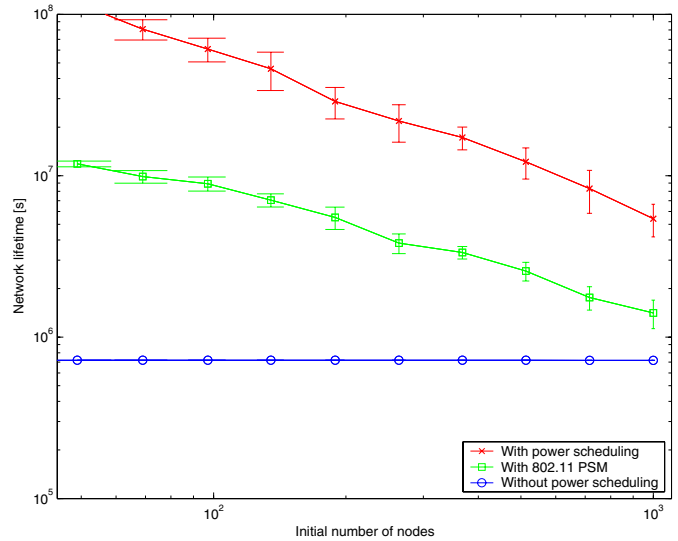


Fig. 7. Dependency of the network lifespans on the number of nodes for a constant density.

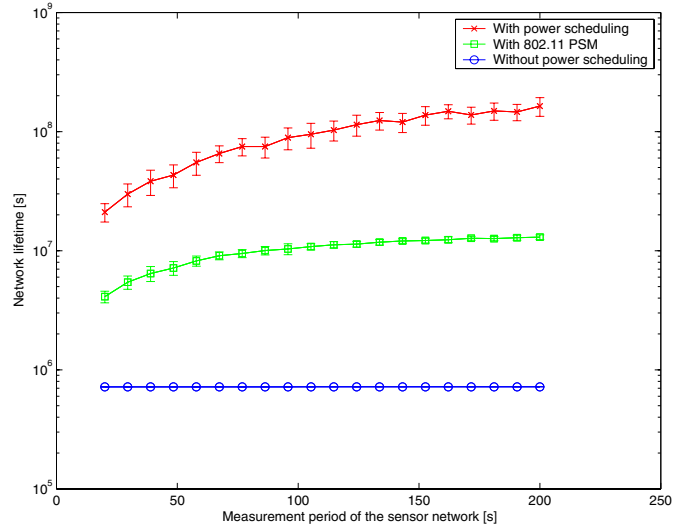


Fig. 8. Dependency of the network lifespans on the measurement period of the network.

enables longer sleep times and, correspondingly, increased improvements in the lifespan of the sensor network. The lifetime of the system without power savings improves only marginally, as the difference between the transmit and receive power does not result in significant power savings.

As expected, the power consumption in idle mode is extremely important for the proposed scheme (see Fig. 9). Essentially, the power savings of the method result directly from the difference between the power of the system in idle mode and the power in receive mode. When the gap between idle mode and receive mode closes, the power savings of the proposed approach decrease. Similar effects occur for 802.11 PSM.

In Section II we briefly discussed how to implement the

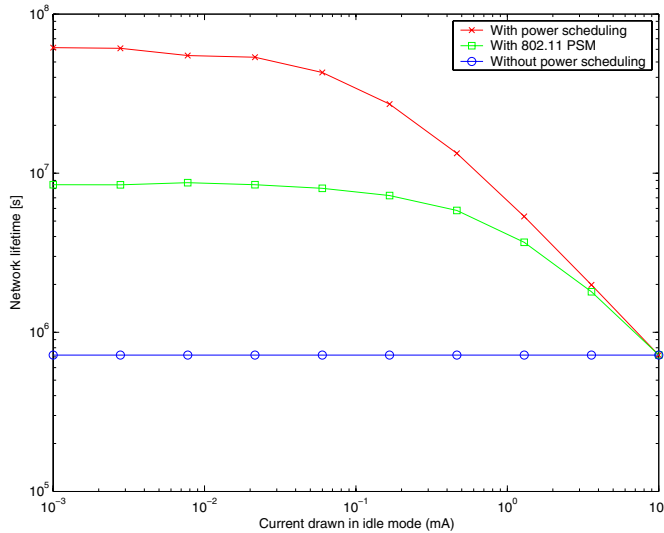


Fig. 9. Dependency of the network lifespans on the power consumption in idle mode.

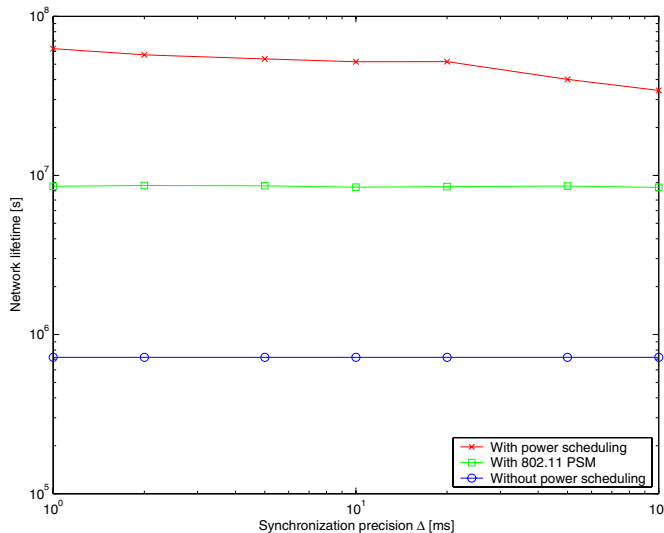


Fig. 10. Dependency of the network lifespans on Δ , the precision of the synchronization algorithm.

proposed approach if a realistic synchronization algorithm (providing finite precision Δ) is employed. Figure 10 shows that even if the precision is poor (100ms), the scheme works quite well (the reduction in the network lifetime is moderate). Significantly better synchronization precisions (tens of microseconds) have been demonstrated in the context of sensor networks [41], [42].

IV. CONCLUSION

We presented a new distributed scheduling algorithm for stationary continuous monitoring sensor networks. The proposed scheme exploits the time scale difference between sensor network reconfiguration periods and data forwarding periods. It is very likely that the approach will not be suitable for event driven sensor networks, where the schedule setup

overhead will likely outweigh the power savings. The approach is fully distributed and works in tight cooperation with popular sensor networks routing and MAC families of protocols. The approach does not fit cleanly in any one layer, as it requires the collaboration of both the routing and MAC layers. For the right type of networks (with long-lived CBR flows and low duty cycles), it is shown via simulations that the network lifetime can be increased by orders of magnitude.

REFERENCES

- [1] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Communication Magazine*, vol. 40, no. 8, pp. 102–116, Aug. 2002.
- [2] —, "Wireless sensor networks: A survey," *IEEE Computer*, vol. 38, no. 4, pp. 393–422, Mar. 2002.
- [3] G. Asada, T. Dong, F. Lin, G. Pottie, W. Kaiser, and H. Marcy, "Wireless integrated network sensors: Low power systems on a chip," in *Proc. of the 24th European Solid-State Circuits Conference*, The Hague, Netherlands, 1998. [Online]. Available: citeseer.nj.nec.com/278712.html
- [4] S. R. Madden, R. Szewczyk, M. J. Franklin, and D. Culler, "Supporting aggregate queries over ad-hoc wireless sensor networks," in *Workshop on Mobile Computing and Systems Applications*, 2002.
- [5] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "TAG: a Tiny AGgregation service for ad-hoc sensor networks," in *Proc. of OSDI*, Dec. 2002.
- [6] S. Madden and M. J. Franklin, "Fjording the stream: An architecture for queries over streaming sensor data," in *Proc. of ICDE 2002*, 2002.
- [7] P. Bonnet, J. Gehrke, and P. Seshadri, "Towards sensor database systems," in *Proc. of the 2nd International Conference on Mobile Data Management*, Hong Kong, Jan. 2001.
- [8] A. Shatdal and J. Naughton, "Adaptive parallel aggregation algorithms," in *Proc. of ACM SIGMOD*, 1995.
- [9] D. L. Tennenhouse, J. M. Smith, W. D. Sincoskie, D. J. Wetherall, and G. J. Minden, "A survey of active network research," *IEEE Communications Magazine*, 1997.
- [10] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "The design of an acquisitional query processor for sensor networks," in *In Proc. of (SIGMOD'03)*, San Diego, CA, June 2003. [Online]. Available: <http://www.cs.berkeley.edu/~madden/acqp.pdf>
- [11] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient MAC protocol for wireless sensor networks," in *Proceedings of the IEEE Infocom*, USC/Information Sciences Institute, New York, NY, USA: IEEE, June 2002, pp. 1567–1576. [Online]. Available: <http://www.isi.edu/~johnh/PAPERS/Ye02a.html>
- [12] D. E. Culler, J. Hill, P. Buonadonna, R. Szewczyk, and A. Woo, "A network-centric approach to embedded software for tiny devices," in *Proc. of the First International Workshop on Embedded Software (EMSOFT)*, Oct. 2001.
- [13] A. Woo and D. E. Culler, "A transmission control scheme for media access in sensor networks," in *Proc. ACM MOBICOM*, July 2001.
- [14] R. Min, M. Bhardwaj, N. Ickes, A. Wang, and A. Chandrakasan, "The hardware and the network: Total-system strategies for power aware wireless microsensors," in *Proc. of the IEEE CAS Workshop on Wireless Communications and Networking*, Pasadena, CA, USA, Sep. 2002. [Online]. Available: <http://www.mit.edu/~rmin/research/min-cas02.pdf>
- [15] R. Ramanathan and R. Hain, "Topology control of multihop wireless networks using transmit power adjustment," in *INFOCOM (2)*, 2000, pp. 404–413. [Online]. Available: citeseer.nj.nec.com/ramanathan00topology.html
- [16] R. Wattenhofer, L. Li, P. Bahl, and Y.-M. Wang, "Distributed topology control for power efficient operation in multihop wireless ad hoc networks," in *Proc. of INFOCOM*, 2001.
- [17] S. Narayanaswamy, V. Kawadia, R. S. Sreenivas, and P. R. Kumar, "Power control in ad-hoc networks: Theory, architecture, algorithm and implementation of the COMPOW protocol," in *Proc. of European Wireless 2002. Next Generation Wireless Networks: Technologies, Protocols, Services and Applications*, Florence, Italy, Feb. 25-28 2002, pp. 156–162. [Online]. Available: http://black.csl.uiuc.edu/~prkumar/ps_files/compow_ewc_2002.pdf

- [18] S. Singh and C. Raghavendra, "Power efficient MAC protocol for multihop radio networks," in *The Ninth IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, 1998, pp. 153–157.
- [19] R. Rozovsky and P. R. Kumar, "SEEDEX: A MAC protocol for ad hoc networks," in *Proc. of The ACM Symposium on Mobile Ad Hoc Networking & Computing, (MobiHoc)*, Long Beach, CA, Oct. 2001, pp. 67–75.
- [20] Y.-C. Tseng, C.-S. Hsu, and T.-Y. Hsieh, "Power-saving protocols for IEEE 802.11-based multi-hop ad hoc networks," in *Proc. of INFOCOM*, vol. 1, 2002, pp. 200–209.
- [21] M. Stemm and R. H. Katz, "Measuring and reducing energy consumption of network interfaces in hand-held devices," *IEICE Transactions on Communications*, vol. E80-B, no. 8, pp. 1125–1131, 1997.
- [22] L. M. Feeney and M. Nilsson, "Investigating the energy consumption of a wireless network interface in an ad hoc networking environment," in *Proc. of IEEE INFOCOM*, 2001, pp. 1548–1557.
- [23] S. Singh and C. Raghavendra, "PAMAS: Power aware multi-access protocol with signalling for ad hoc networks," *ACM Computer Communications Review*, 1999. [Online]. Available: citeseer.nj.nec.com/460902.html
- [24] E.-S. Jung and N. Vaidya, "A power saving mac protocol for wireless networks," Technical Report, UIUC, July 2002.
- [25] S. Singh, M. Woo, and C. S. Raghavendra, "Power-aware routing in mobile ad hoc networks," in *Mobile Computing and Networking*, 1998, pp. 181–190. [Online]. Available: citeseer.nj.nec.com/singh98poweraware.html
- [26] M. W. Subbarao, "Dynamic power-conscious routing for MANETS: an initial approach," in *IEEE Vehicular Technology Conference*, 1999, pp. 1232–1237.
- [27] J. Aslam, Q. Li, and D. Rus, "Three power-aware routing algorithms for sensor networks," *Wireless Communications and Mobile Computing*, vol. 2, no. 3, pp. 187–208, Mar. 2003.
- [28] I. Stojmenovic and X. Lin, "Power-aware localized routing in wireless networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 12, no. 11, pp. 1122–1133, 2001. [Online]. Available: citeseer.nj.nec.com/stojmenovic00poweraware.html
- [29] J.-H. Chang and L. Tassiulas, "Energy conserving routing in wireless ad-hoc networks," in *INFOCOM (I)*, 2000, pp. 22–31. [Online]. Available: citeseer.nj.nec.com/chang00energy.html
- [30] Y. Xu, J. Heidemann, and D. Estrin, "Adaptive energy-conserving routing for multihop ad hoc networks," USC/Information Sciences Institute, Research Report 527, October 2000. [Online]. Available: <http://www.isi.edu/johnh/PAPERS/Xu00a.html>
- [31] T. A. ElBatt, S. V. Krishnamurthy, D. Connors, and S. Dao, "Power management for throughput enhancement in wireless ad-hoc networks," in *IEEE International Conference on Communications*, 2000, pp. 1506–1513.
- [32] J. P. Monks, V. Bhargavan, and W.-M. W. Hwu, "A power controlled multiple access protocol for wireless packet networks," in *Proceedings of INFOCOM*, 2001, pp. 219–228.
- [33] J.-C. Cano and P. Manzoni, "Evaluating the energy-consumption reduction in a manet by dynamically switching-off network interfaces," in *Proc. of the Sixth IEEE Symposium on Computers and Communications*, 2001, pp. 186–191.
- [34] R. Kravets, K. Schwan, and K. Calvert, "Power-aware communication for mobile computers," in *Proc. of the International Workshop on Mobile Multimedia Communications (MoMuc'99)*, 1999. [Online]. Available: citeseer.nj.nec.com/kravets99poweraware.html
- [35] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris, "Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks," in *MOBICOM 2001*, July 2001.
- [36] E. Shih, S.-H. Cho, N. Ickes, R. Min, A. Sinha, A. Wang, and A. Chandrakasan, "Physical layer driven protocol and algorithm design for energy-efficient wireless sensor networks," in *Proc of Mobicom 2001*, 2001.
- [37] IEEE, "Wireless LAN medium access control (MAC) and physical layer (PHY) specification," IEEE Std. 802.11, June 1999.
- [38] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. E. Culler, and K. S. J. Pister, "System architecture directions for networked sensors," in *Architectural Support for Programming Languages and Operating Systems*, 2000, pp. 93–104. [Online]. Available: citeseer.nj.nec.com/382595.html
- [39] A. Mainwaring, J. Polastre, R. Szewczyk, and D. Culler, "Wireless sensor networks for habitat monitoring," in *Proc. of the First ACM International Workshop on Wireless Sensor Networks and Applications*, Atlanta, GA, Sept. 2002.
- [40] A. Cerpa, J. Elson, D. Estrin, L. Girod, M. Hamilton, and J. Zhao, "Habitat monitoring: Application driver for wireless communications technology," in *Proc. of the ACM SIGCOMM Workshop on Data Communications in Latin America and the Caribbean*, 2001.
- [41] J. Elson, L. Girod, and D. Estrin, "Fine-grained network time synchronization using reference broadcasts," in *UCLA Technical Report 020008*, Feb. 2002. [Online]. Available: citeseer.nj.nec.com/elson02finegrained.html
- [42] J. Elson and D. Estrin, "Time synchronization for wireless sensor networks," in *Proc. of the 2001 International Parallel and Distributed Processing Symposium (IPDPS), Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing*, San Francisco, CA, Apr. 2001.
- [43] K. Römer, "Time synchronization in ad hoc networks," in *Proc. of ACM Mobihoc*, Long Beach, CA, 2001.
- [44] M. L. Sichitiu and C. Veerarittiphan, "Simple, accurate time synchronization for wireless sensor networks," in *IEEE Wireless Communications and Networking Conference (WCNC 2003)*, New Orleans, LA, Mar. 2003. [Online]. Available: <http://www4.ncsu.edu:8030/mlsichit/Research/Publications/synchro.pdf>
- [45] P. Gupta and P. R. Kumar, "The capacity of wireless networks," *IEEE Trans. on Information Theory*, vol. 46, no. 2, Mar. 2000.
- [46] D. Estrin, R. Govindan, J. S. Heidemann, and S. Kumar, "Next century challenges: Scalable coordination in sensor networks," in *Mobile Computing and Networking*, 1999, pp. 263–270. [Online]. Available: citeseer.nj.nec.com/article/estrin99next.html
- [47] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion: a scalable and robust communication paradigm for sensor networks," in *Mobile Computing and Networking*, 2000, pp. 56–67. [Online]. Available: citeseer.nj.nec.com/intanagonwiwat00directed.html
- [48] J. S. Heidemann, F. Silva, C. Intanagonwiwat, R. Govindan, D. Estrin, and D. Ganesan, "Building efficient wireless sensor networks with low-level naming," in *Symposium on Operating Systems Principles*, 2001, pp. 146–159. [Online]. Available: citeseer.nj.nec.com/heidemann01building.html
- [49] H. Xiaoyan, X. Kaixin, and M. Gerla, "Scalable routing protocols for mobile ad hoc networks," *IEEE Network*, vol. 16, no. 4, July-Aug 2002.
- [50] E. Royer and C. Toh, "A review of current routing protocols for ad-hoc mobile wireless networks," *IEEE Personal Communications*, Apr. 1999. [Online]. Available: citeseer.nj.nec.com/royer99review.html
- [51] J. Raju and J. Garcia-Luna-Aceves, "A comparison of on-demand and table-driven routing for ad-hoc wireless networks," in *Proc. of IEEE ICC*, June 2000.
- [52] V. Bhargavan, A. J. Demers, S. Shenker, and L. Zhang, "MACAW: A media access protocol for wireless LAN's," in *SIGCOMM*, 1994, pp. 212–225. [Online]. Available: citeseer.nj.nec.com/bhargavan94macaw.html
- [53] C. L. Fullmer and J. J. Garcia-Luna-Aceves, "Floor acquisition multiple access (FAMA) for packet-radio networks," in *SIGCOMM*, 1995, pp. 262–273. [Online]. Available: citeseer.nj.nec.com/fullmer95floor.html
- [54] K. Xu, M. Gerla, and S. Bae, "How effective is the IEEE 802.11 RTS/CTS handshake in ad hoc networks?" in *Proc. of IEEE Globecom 2002*, Taipei, Taiwan, R.O.C., Nov. 17-21 2002.
- [55] "Self discharge of batteries," <http://www.voltaicpower.com/Batteries/self-compare.htm>.

APPENDIX

In this appendix, we sketch the pseudo-code of the algorithm that each node follows when scheduling, then servicing (i.e., forwarding the data) one flow. Ten consecutive packets are an indication that the flow has been interrupted, and an error is sent toward the source. We assume that the routing protocol determined for each node a list of next-hop neighbors that have a lower hop count than the current node. We reserve the slot at x.000 for control messages and retransmissions. This is a bare-bones pseudocode that does not enter into the details of the implementation.

```
const MAX_MISSED_PACKETS = 10;

schedule_table
(Action, Time, SourceNode, previous_hop, next_hop,
 active_flow)

enter_action_in_schedule_table(
 Action=listen_for_control_packets,
 time=x.000,
 nil,nil,nil)

temporary_schedule_table
(Action, Time, SourceNode, previous_hop, next_hop)

switch(event) {

case: received RSETUP packet
 enter new receive entry in the schedule table;
 next_neighbor = first neighbor with
 smaller hop count than self.
 generate event 'transmit RSETUP packet';

case: transmit RSETUP packet
 setup timer timer_abort to fire after one period;
 append own address to RSETUP packet;
 while (1)
 try to transmit RSETUP packet to
 neighbors(next_neighbor)
 if successful{
 cancel timer_abort;
 enter transmit entry into the temporary schedule table;
 break;
 }

case: (received RERR) or (timer_abort fired)
 //i.e., unsuccessful schedule to this neighbor
 if no more neighbors
 delete schedule entry from temporary schedule
 if SourceNode == self
 Give up // the flow cannot be scheduled
 else
 send RERR toward SourceNode
 else
 try the next neighbor with smaller hop count than self
 generate event 'transmit RSETUP packet';

case: received RACK packet
 move the two schedule entries from
 the temporary table to the schedule table
 active_flow=MAX_MISSED_PACKETS;
 forward RACK to the previous hop.
 mark flow as

case: received query packet
 insert a sampling action into the schedule table
 next_neighbor=first neighbor with smaller hop count than self.
 generate event 'transmit RSETUP packet';

case: Sampling is scheduled
 take data sample;
 go to sleep;

case: Reception is scheduled for flow from SourceNode
 receive(packet);
 if packet is valid{
 active_flow=MAX_MISSED_PACKETS;
 put the packet in the output buffer
 }
 else{
 active_flow--;
 if active_flow==0{
 purge both entries from SourceNode from the schedule table
 send RERR to the source node
 }
 }
 goto sleep;

case: Transmission is scheduled for flow from SourceNode
 if packet present in the output buffer,
 forward packet to nextHop

case: listen for control packets
 put transceiver in receive mode and listen for control packets
}
}
```