Power-Efficient Data Dissemination in Wireless Sensor Networks*

Uğur Çetintemel
Brown University
Dept. of Computer Science
Providence, RI 02912
ugur@cs.brown.edu

Andrew Flinders
Brown University
Dept. of Computer Science
Providence, RI 02912
awf@cs.brown.edu

Ye Sun
Brown University
Dept. of Computer Science
Providence, RI 02912
ys@cs.brown.edu

ABSTRACT

This paper presents a new event-based communication model for wireless multi-hop networks of energy-constrained devices such as sensor networks. The network is arranged as an event dissemination tree, with nodes subscribing to the event types they are interested in. An event scheduler dynamically allocates and multiplexes upstream and downstream time slots for each event type. Power consumption among wireless nodes is reduced by allowing each node to power down its radio during the portions of the schedule that do not match its particular event subscription. The event dissemination schedule can be determined in both a centralized and distributed fashion, and is highly dynamic to suit the changing rates at which events are generated and distributed through the network. The paper also presents preliminary performance results that demonstrate the power savings achieved by the proposed protocols.

Categories and Subject Descriptors: H.3.4 [Systems and Software]: Distributed Systems

General Terms: Algorithms, Performance.

Keywords: Wireless Networks, Sensor Networks, Scheduling, Topology Management, Publish/Subscribe.

1. INTRODUCTION

The confluence of ubiquitous RF-based wireless networking [14, 4, 3] and recent advances in low-power analog and digital electronics has led to the emergence of compact, inexpensive, battery-operated sensor units equipped with computational and wireless communication capabilities [22, 33]. Due to their increasingly favorable form and cost factors, it is feasible to link together a large number of such sensors in order to support fault-tolerant, fine-grained monitoring and tracking applications [13, 32]. Cheap and ubiquitous platforms of networked sensors will be the key to real-time delivery of large volumes of useful information and will support a variety of civil and military applications (such as traffic monitoring, disaster recovery, industrial tracking, factory instrumentation, inventory tracking, battlefield monitoring, and intrusion detection).

Many current sensor networks consist of a relatively small number of sensor units that are typically wired to a central processing system. In constrast, in many emerging and envisioned applications, sensor networks will be both *distributed* and *wireless* (in terms of communication and power) [10]. Distribution is necessary for improving sensing quality: when the precise location of a signal is unknown, then distributed sensors will allow sensing to take place closer to the event of interest than what is possible by any single sensor. Distribution also improves robustness to environmental obstacles, which is especially crucial in situations where sensing requires line-of-sight. Furtermore, the target environments typically lack already existing communications or energy infrastructures. The sensor units, thus, need to rely on finite, local energy sources and wireless communications channels. Finally, shorter-range communication is generally much cheaper than longer-range communication because the radio-signal power drops off with the fourth power of the distance [13, 25]. As a result, it is much cheaper to transmit information using multi-hopping among sensor units.

Consider the following application scenarios for distributed wireless sensor networks. *Habitat monitoring* applications involve data collection from and modeling of complex ecosystems without disturbing the habitat and wildlife. Monitoring facilities must be deployed in remote locations that do not have pre-existing communication and energy resources. As a case in point, researchers have recently deployed wireless sensor networks on Great Duck Island, Maine, that monitor the microclimates in and around nesting burrows used by Leach's Storm Petrel [2]. Secondly, consider an *ad hoc smart space* where large numbers of disposable sensors that are densely scattered into a building damaged by an earthquake [12]. The sensors can establish an ad hoc communication network and cooperate to divide the task of mapping the structural damage in an efficient manner. Finally, *tactical operations* taking place in unknown or hostile regions can also benefit from ad hoc wireless sensor networks. Soldiers can be equipped with battery-operated sensors that monitor critical signs such as body temperature or heart rate. In case of an emergency situation, the wireless network automatically directs the nearest emergency response team to wounded individuals [31].

In these and similar sensor network applications involving battery-operated nodes, network longevity (i.e., total opera-

* This work is partially supported by a gift from Microsoft Corporation.

tional lifetime) is a key consideration. In many cases, replacing or recharging batteries may be impractical or uneconomical. For instance, during a tactical mission, it is clearly not desirable to have to be concerned about replacing batteries (which requires human interference and attention). Another example involves next-generation smart dust-style sensor networks [22], which are composed of densely populated units that are deeply embedded into the environment. The smart dusts are expected to be so cheap that they will be practically disposable: it will make more economical sense to incrementally deploy new nodes rather than to replace the batteries of the existing ones.

Designing energy-efficient systems is a research goal of critical importance for a variety of networking domains, including sensor networks and mobile ad hoc networks [36]. The primary consumer of energy in wireless networks is communication. RF communications are very costly compared to other electrical hardware functions including instruction execution. Based on this observation, many researchers have studied energy-minimization techniques that reduce communication at the expense of extra computation. Most work focused on developing approaches that reduce the volume of data that need to be transmitted, typically through intelligent data reduction and aggregation techniques (e.g., [18, 26, 27]). Less studied are techniques that enable nodes to power down their antennas (i.e., go to *sleep* or *standby* mode) during times of inactivity. This latter set of techniques are particularly promising as the energy consumed by short-range RF communications in transmission (Tx) and listening (Rx) modes are quite similar for many existing wireless hardware and protocols (including 802.11 [3] and Bluetooth [14]). As a result, the only way to significantly save energy is to completely turn the radio off [34, 38]. This paper presents an integrated data scheduling and dissemination protocol that leverages this key fact.

The proposed protocol, called *Topology-Divided Dynamic Event Scheduling* (TD-DES), organizes the wireless network into a multi-hop network tree. The root of the tree creates a *data dissemination schedule* and propagates this schedule throughout the tree, so that all nodes may adhere to it. The schedule is divided into fixed-size time slots, each indicating the type of data that are sent (or received), and whether it is for *downstream* (i.e., away from the root) or *upstream* (i.e., toward the root) communication. The schedule can be periodic or refreshed in arbitrary intervals, depending on the data traffic and applications. In either case, the central idea is that nodes can save energy by powering down their radios to standby mode when they have no data to send, and when they (and their descendants) do not wish to receive the data being transmitted. The system uses the *publish/subscribe* model: each node has a specific subscription profile that indicates which data types the node is interested in receiving.

In a traditional wireless network, nodes must listen promiscuously to the wireless channel for all data being transmitted, lest they might miss something important. TD-DES allows each node to selectively listen for data which interests it, based on the its position in the network topology, and save energy otherwise. Because data must be scheduled before it is sent, the main tradeoff that we investigate is increased power efficiency in exchange for sub-optimal message dissemination latency.

Significant work has been done in the area of wireless broadcast scheduling. Early work (e.g., [17, 7]) addressed power consumption reduction in flat broadcast environments and considered only asymmetric, downstream-only data dissemination. More recent work [38, 34] attacked the same problem in wireless, multi-hop network environments but did not address application-specific scheduling and data dissemination issues, which constitute the central theme of this paper (see Section 5 for a detailed discussion of related prior work).

The rest of the paper is structured as follows. Section 2 describes the basic sensor network model referenced throughout the rest of the paper. Section 3 introduces and describes in detail our integrated scheduling and data dissemination protocols. Section 4 provides the results of a preliminary characterization of the performance of our protocol. Section 5 describes related work, and Section 6 provides concluding remarks and outlines directions for future research.

2. SYSTEM MODEL

This paper presents TD-DES, a data event scheduling and dissemination protocol specifically designed for wireless multi-hop networks of energy-constrained devices. TD-DES is intended as an application overlay to a CSMA/CA wireless MAC layer (such as 802.11 MAC DCF [6] and MACA [24]), rather than a MAC/networking layer in itself.

2.1 Scheduling Model

TD-DES's main function is to govern when each node of a network (1) receives data, (2) transmits data, and (3) powers its radio down to a low-power standby mode. These radio modes – Tx, Rx, and standby – are cycled among as functions of time determined by the network's *dissemination schedule*, which is generated primarily by the *root node* and propagated down the tree as part of a *control event*. The root node is assumed to be a *base station* with greater computational, storage, and transmission capabilities than the rest of the nodes in the network. The root node typically serves an entry point to the sensor network, integrating the sensor network with the external wired network where the monitoring task GUI resides. As we will describe in Section 3, the scheduler relies on topology information, event profiles, traffic statistics, and Quality-of-Service (QoS) expectations when generating dissemination schedules. The basic goal of the scheduler is to minimize network-wide power consumption (by minimizing the amount of time spent in the Rx and Tx modes) without sacrificing timely dissemination of data.

2.2 Network Model

TD-DES also provides an integrated network construction layer, which organizes a wireless network into a tree topology. The topology is constucted by broadcasting advertisements from all nodes in the network. Initially the root node broadcasts a *parent* advertisement. Each node that hears the advertisement replies with a *child* message that indicates that the node is willing to become a child of the root. Whenever a node becomes a child, it broadcasts its own parent advertisement. The process continues until all the nodes get attached to the tree. A node that hears multiple parent advertisements chooses as its parent the node with the lowest hop count to the root. This style of tree construction is not novel and has been commonly used by other work [5, 16, 26].

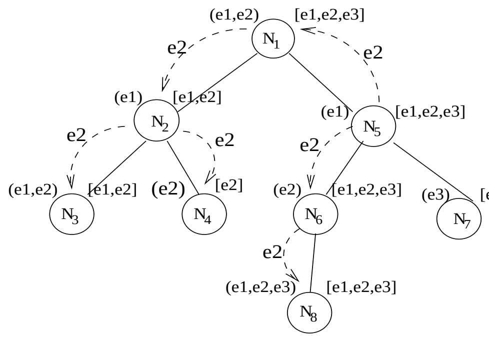

Figure 1: An example dissemination tree. Subscriptions are given at the upper left of each node, effective subscriptions at the upper right. Arrows indicate the links over which the event is broadcast.

The tree construction layer is adaptive to changes in the topology that result from node failures, additions, and mobility. The layer does not provide point-to-point messaging capabilities (e.g., using globally unique id's such as IP addresses) as do traditional routing protocols [21, 29, 30], but rather provides a means of disseminating data events throughout the network to all interested parties based on per-node event subscriptions. This *publish/subscribe* style of event-based communication is the data dissemination model of choice because it effectively decouples the producers and consumers of information, making it well-suited for dynamic, ad-hoc environments.

2.3 Data/Event Model

Predefined event types are defined by the overlaying applications and are maintained in a global event schema. In particular, a network with n different event types may publish event types $e_1, e_2, ..., e_n$. Each node maintains its own *event subscription*, which is the set of event types that a node is interested in receiving. In addition, each node maintains its own *effective subscription*, which is the union of its own subscription and the subscriptions of all of its descendents [16]. Each node must effectively subscribe to any event type that it itself is interested in as well as any event type that a descendent node is interested in. This is because each node is responsible for *forwarding* all relevant events to its descendants in the tree topology.

Figure 1 illustrates a dissemination tree of eight nodes and three event types e_1, e_2, and e_3. N_1 is the root node of the tree. The subscription of each node is given in parentheses at the upper left of the node. The effective subscription of each node is given at the upper right of each node in square brackets. The figure also shows an event of type e_2 generated at node N_5. The arrows indicate the links across which the event is broadcast in order to disseminate the event to all subscribing nodes. Note that the event is propagated both upstream (to the root and then downstream to the interested parties in the other subtree) and downstream. As a result, events do not necessarily always go through the root node.

In general, an event is a particular message type with its own distinguishing, application-specific semantics. Consider a scenario where a sensor network whose purpose is to detect fires is deployed over a forested region. A sensor node might issue a **fire_detected** event to the network if its thermal sensor registered a very high temperature reading. The event would be disseminated through the network to all those nodes subscribing to **fire_detected** events. These could include nearby forest ranger stations, a centralized forest fire monitoring station, or a sink node which could notify the police, local fire-fighting units, and public news services. These would also include any intermediate nodes which had to *forward* such events to interested nodes, even if they themselves were not interested per se. A different type of event could be a **low_battery** event, which would only be disseminated to a network maintenance facility, alerting personnel that a particular sensor node would soon fail if its battery was not quickly replaced. Another type of event could alert the same facility that the sensor equipment on a node was not functioning properly.

2.4 Application-defined Quality-of-Service

Besides carrying distinguishing type semantics (with which individual network nodes can decide whether or not to include them in their subscriptions), event types may also be associated with network-specific physical characteristics, such as minimum and maximum event payload sizes (in bytes), latency constraints (e.g., maximum allowable propagation delay), and relative event priorities. TD-DES allows such event latency and priority values to be specified by the overlaying applications. Such parameters affect the ordering policies of the event scheduling algorithm (Section 3). However, for maximum interoperability with existing applications, the system also works in the absence of such parameters.

3. PROTOCOLS

The TD-DES event schedule determines the temporal partitioning of the RF medium for all of the event types in the system. This is accomplished by allocating *time slots* (or *slots* for short) for each event type (subsection 3.1). Each time slot is assumed to be wide enough (i.e., provides roughly enough time) for a single event to be propagated one hop. Put another way, each slot should be wide enough to provide sufficient time to the underlying MAC layer to perform collision detection and retransmissions under contention.

Time slots are allocated for each event based on the *determined* or *expected* bandwidth requirements needed to propagate all generated events reliably thoughout the network (subsection 3.2). Once the numbers of upstream and downstream time slots for each event type are determined, the ordering of the time slots must then be determined (subsection 3.3). Iterations are intervals of schedule that starts with a control event slot. It is also possible to interleave downstream and upstream slots together to fit into a single iteration (subsection 3.4).

3.1 Schedule Propagation

The root node is responsible for creating a schedule of time slots. Each time slot is designated as a send or receive slot, whether it is for upstream or downstream communication, and by the event type which it should be used to propagate. The root creates the schedule one iteration at a time and passes it down through the dissemination tree inside a control event. The schedule of slots between two consecutive downstream control events is called a single *iteration* of the schedule.

Figure 2 illustrates the basic idea of creating a schedule

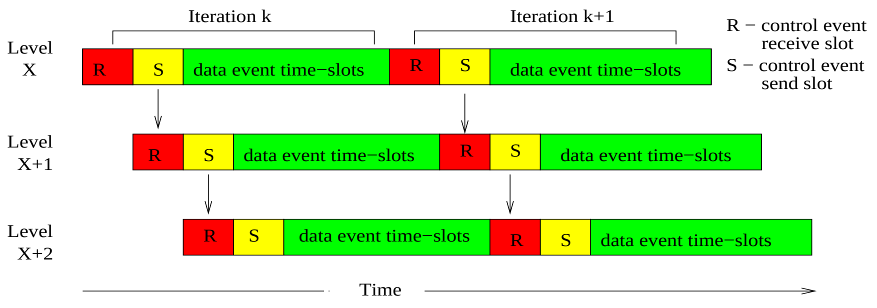

Figure 2: Example of schedule propagation

and passing it down the network tree using a simple scenario that involves downstream propagation of control and data events. The control event is a special type of event which is created by TD-DES and contains scheduling information. The control event is received by the node at level X in the first time slot. In the next time slot, this node transmits the control event down to the next level, $X+1$. In the time slot following this, the node at level $X+1$ passes the control event down to level $X+2$, and so on and so forth. To summarize, iterations are delimited by control events and can consist of a different number of data events. The control event that initiates an iteration specifies the schedule of events (which events will be disseminated and when) within that iteration. Note that the schedule is shifted one slot at each level.

The schedule consists of a sequence of atomic *send* and *receive time slots*, each one for a specified event type. In general, for a particular event in a schedule, at a given node, time slots are allocated as a receive slot followed by an immediate send slot most of the time. At a particular node, if the schedule specifies a receive time slot for event e_1, the node, if it subscribes to e_1, will listen to the RF medium in Rx mode during this time slot to receive such an event. If the schedule specifies a send time slot for e_1, the node can use this time slot to transmit an event of this type (should it have one to send).

Furthermore, each slot is distinguished as either a *downstream* slot (for parent-to-child communication away from the root) or as an *upstream* slot (for child-to-parent communication toward the root). For downstream communication, *send* and *receive* slots are used. Upstream slots are not designated for event types, as they are speculatively allocated – any event which is generated should be able to make use of the next upstream slot. In addition, nodes must always listen to upstream receive slots, as all events must be passed up to the root, regardless of event type. Therefore, distinguishing upstream slots for particular event types would serve no beneficial purpose.

The downstream control event also includes data used by the tree construction protocol, such as the number of hops to the root and the parent node's network-unique identifier. Notice that for each downstream *send* event, the simultaneous time slot at the next level down is a *receive* time slot for the same type of event. Similarly, for upstream *send* events, the concurrent time slot at the next level up the tree is a corresponding *receive* time slot for the same type of event.

3.2 Deterministic and Speculative Scheduling

TD-DES can schedule time slots in two modes: *deterministic* and *speculative*. We use the deterministic algorithm for downstream and the speculative algorithm for upstream dissemination. The decision is based on the observation that most event propagation would be downstream (even when events are generated at internal nodes).[1]

In the deterministic algorithm, events are propagated in back to back iterations. Each iteration is further divided into slots of fixed width. The scheduler (root node), already knows the exact events to be broadcast at the beginning of each iteration and allocates the exact number of slots required. The schedule is propagated to every node in the tree in the form of a control packet at the beginning of each iteration. In addition to containing the schedule for events, a control packet can also contain timing information for the next control packet, if iterations are not of fixed length. When the root node starts trasmitting events, each node just needs to leave radio in Rx mode for the duration of the slot when some interesting event will arrive.

Figure 3 illustrates the process of deterministic scheduling. As before, R and S denote the receive and send slots for the control events, respectively. Event e_1 generated during iteration k cannot be scheduled till iteration $k+1$. The control event transmitted during the second S includes the schedule for iteration $k+1$. The exact time slot during which e_1 will be scheduled is determined by the specific ordering criterion used.

In speculative scheduling, the scheduler estimates the (expected) frequency of event types at the root node and preallocates slots based on this combined frequency estimation. Since allocation of slots for each event type is *periodic* and therefore is the same from one iteration to the next, no schedule broadcasting is needed except when updating schedule. The disadvantage of speculative scheduling is that nodes might have to stay in Rx mode for scheduled slots regardless of whether or not interesting event is actually coming.

Figure 4 illustrates the process of speculative scheduling. Event e_1 is received during iteration k after its scheduled slot (indicated by the dashed lines). Thus, e_1 needs to be queued before it can be transmitted during its slot in iteration $k+1$.

Regardless of which algorithm is being used, the schedule decided by the root node is known to every node in the tree. A child node's downstream schedule is one slot behind its parent node's downstream schedule. A child node's upstream schedule is one slot ahead of the parent node's upstream schedule. This allows tight pipelining: a downstream/upstream event received by node i in slot t will be

[1]We note that our scheduling mechanism is general and does not in any way require upstream scheduling to be speculative.

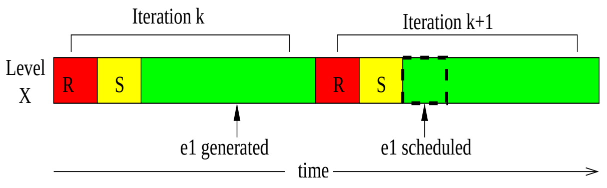

Figure 3: Deterministic scheduling

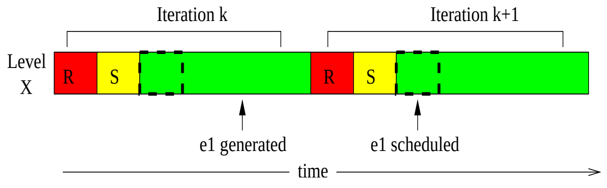

Figure 4: Speculative scheduling

sent downward/upward to i's children/parent in slot $t+1$. If shifting happens at the boundary of upstream and downstream schedule, downstream scheduling will shift beyond the neighboring upstream schedule and upstream scheduling will shift beyond the neighboring downstream schedule.

The schedule as determined by the root node can be extended at an internal node to accomodate events generated at internal nodes. The schedule decided by the root node has to allocate slots for all events. A subtree rooted at an internal node might not be interested in every event. So when internal node is propagating down root schedule to its descendants, it can extend the root schedule by replacing those *un-interesting* slots with slots for its own events or if even more slots are required, it can modify blank slot in the root schedule. Note that this extended schedule only affects the subtree rooted at this internal node.

3.3 Scheduling Criteria

In this section, we focus on how the root node decides on event ordering in the downstream schedule. Once the iteration length and slot length become fixed, a deterministic schedule is just an ordering of events. We consider ordering events according to one of (or combination of) three criteria:

- *priority* - the relative priority of an event type over other event types
- *popularity* - the number of nodes subscribing to an event type
- *latency constraint* - the maximum allowable dissemination delay for an event type

Priorities can be specified by the application-layer for event types at the root node and passed down the tree in the body of the *downstream control event.* If the priorities are relatively fixed, they need only be included in the control event on occasion (as often as new event types are added or the priorities change).

We can also order events by *popularity.* In essence, this method assumes that the event types that are most subscribed to are considered to be most important by the system, and so they are scheduled first in the upcoming iteration(s). The tree-construction and maintanance layer of TD-DES gathers the popularity of each event type (and more generally *all* of the network subscription data) in a bottom up fashion. Assume we are looking at subscription to a particular type of event e_i. Each node p maintains a local variable $count(e_i)$ indicating how many nodes in its subtree are interested in this event of type e_i. Using subscript to indicate location of the variable, $count_p(e_i)$ can be computed recursively by

$$count_p(e_i) = \sum_{q \in child(p)} count_q(e_i) + \{0|1\} \qquad (1)$$

where 1 is for the case p itself is subscribed; 0 indicates otherwise.

If *latency constraints* are specified by the application layer for event types, TD-DES will make use of the average- and worst-case latency dissemination estimates when scheduling events. A node can reduce the overall dissemination latency of an event by scheduling it as early as possible – this reduces the scheduling delay component of the latency. When latency is the primary sorting criterion, the scheduler will attempt to schedule events such that their average and worst-case latencies remain within the given maximum bounds. Since no real latency is available at time of scheduling, hop-count-based distance is used as an estimator instead. The number of hops from root for a node k subscribing to event type e_i is called the *distance* of e_i at node k. The average distance for all nodes subscribing to event type e_i is denoted by $distance_{avg}(e_i)$. The worst-case distance is denoted by $distance_{wst}(e_i)$.

In order to determine these event type metrics, the tree gathers data in a bottom-up fashion by having each internal node maintain partial values for its own subtree and pass these values up to its parent node in its *upstream control event.* In particular, each node j maintains the following metrics in addition to $count(e_i)$, which it passes to its parent:

- $cost_j(e_i)$ – the total number of hops that an event e_i must be propagated to the entire subtree rooted at the current node j.
- $avg\_cost_j(e_i)$ – the average number of hops that an event e_i must be propagated per interested node inside the subtree rooted at the current node j.
- $max\_cost_j(e_i)$ – the maximum number of hops that an event e_i must be propagated to an interested node inside the subtree rooted at the current node j.

Each node j passes its $cost_j(e_i)$, and $max\_cost_j(e_i)$ values for each event type e_i to the parent as parameters of its *upstream control event.* For each child j of a particular internal node k, the parent node calculates its own values recursively as follows: The $cost_k(e_i)$ at k is calculated in terms of each child as follows:

$$cost_k(e_i) = \sum_j count_j(e_i) + \sum_j cost_j(e_i) + (0/1) \qquad (2)$$

The maximum cost value is simply the maximum of the maxima of its children plus 1:

$$max\_cost_k(e_i) = \max_j(max\_cost_j(e_i)) + 1 \qquad (3)$$

At each node, the $avg\_cost_k(e_i)$ is a derived value of $count_k(e_i)$ and $cost_k(e_i)$:

$$avg\_cost_k(e_i) = count_k(e_i)/cost_k(e_i) \qquad (4)$$

Finally, the root node, denoted r, then defines, for each event type e_i, the system-wide count and distance values in the following way as: $count(e_i) = count_r(e_i)$, $distance_{avg}(e_i) = avg\_cost_r(e_i)$, and $distance_{wst}(e_i) = max\_cost_r(e_i)$.

Because all internal nodes are interested in knowing these three values (for scheduling by latency or popularity), the root node disseminates these values in a *downstream control event* as often as they change.

3.4 Interleaved Scheduling

The last stage of the TD-DES scheduling algorithm involves determining the actual sequence of the time slots allocated for the next iteration. At this point, the number and ordering of events in downstream schedule is complete, as well as number of slots in upstream schedule. From these two schedules, the sequencer must derive a set of ordered slots for the next iteration. There are two choices here: either place upstream and downstream slots separately side by side (a.k.a. *clustered*) or interleave them.

In the clustered version, the ordered downstream set is placed unbroken, followed immediately by ordered upstream set. These are lastly followed by some *blank* time slots. Additionally, a downstream control event is placed at the beginning of each iteration. The details of interleaving are skipped due to space limitations.

4. EVALUATION

We built an ad-hoc networking simulator to characterize the behavior of two data dissemination models. The first model implemented a CSMA/CA MAC layer (with exponential random backoff and RTS/CTS) along with the TD-DES scheduling overlay and tree construction protocol. The second model, which emulates conventional protocols, implemented the same MAC layer and a similar wireless tree construction protocol, but without TD-DES. Rather than being scheduled, events in the latter network are propagated immediately after generation. We refer to this second network of each pair as the "non-scheduled" network. Without a schedule to follow, nodes in this network must listen "promiscuously," always in Rx mode (except when transmitting). They cannot make use of the standby modes of their tranceivers.

For comparison, both networks were identical in their topologies and event subscription profiles. The same poisson event generation distribution was used to simulate events being generated and disseminated. Topologies were static and all nodes had omnidirectional, uniform broadcast ranges (i.e., links were symmetric)

Figure 5 graphs the relative power consumption of the non-scheduled network to TD-DES as a function of the minimum iteration length (i.e., the minimum number of time slots between two control events). The network consists of 20 nodes; there are three event types; and each node has a 0.2 probability of subscribing to each event type. The time slot width ratio was one – meaning that time slots were exactly the required width to propagate one event (i.e., 2 ms). The radio hardware was modeled after the Proxim RangeLAN2 2.4 GHz 1.6 Mbps PCMCIA card, which expends 1.82W in transmit mode, 1.80W in receive mode, and 0.18W in standby mode [20]. Each node had a radio range of 30 meters and area range was varied from 200 square meters. There are five event generation rates (or EGRs), where an EGR is defined as the average number of time slots between

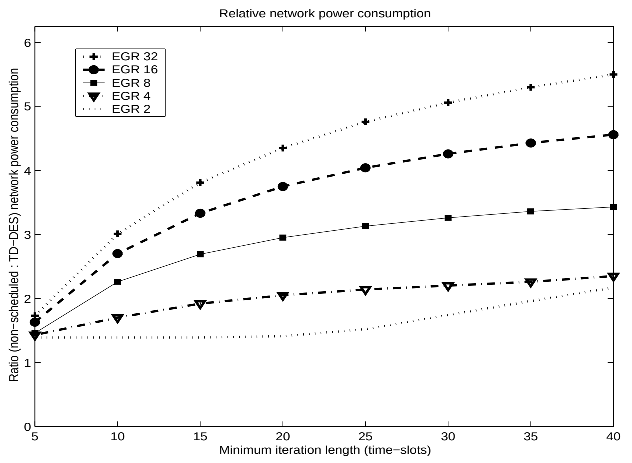

Figure 5: Power consumption ratio (non-scheduled network vs. TD-DES) as a function of iteration length (20 nodes, 0.2 subscription ratio).

events, used by the poisson event generation distribution.

The graph shows that the non-scheduled network consumed anywhere from 142% up to 550% as much as its counterpart TD-DES network. We also observe that the relative power consumption increases with decreasing event-generation frequency. This is because TD-DES can best leverage its power-saving abilities when the traffic load is light. With heavy traffic, nodes cannot utilize their low-power standby modes (as they are nearly always receiving or transmitting data), so the relative power consumption approaches one. Furthermore, as the minimum iteration lenght increases, the power savings also increase because the nodes typically need to listen fewer control messages.

For the same set of results, average dissemination latencies for events were also recorded (but not shown due to space considerations). Dissemination latency is the total time that expires between when an event is generated and when it is received by an interested node. This is the sum of two components: the scheduling delay and the propagation delay. The non-scheduled network experiences no scheduling delay – that is, events are propagated almost immediately after generation. The only source of delay before propagation would be from queueing (if events are generated faster than the rate at which they can be propagated one hop) or medium contention. We observed that the TD-DES network consumed anywhere from 250% up to 1100% as much delay as its counterpart non-scheduled network did. We also observed that the latencies increase with increasing minimum iteration lenght.

These results reveal that TD-DES trades off dissemination latency for power savings. As a result, TD-DES may not be appropriate for real-time applications where dissemination latencies should be kept minimal (on the order of milliseconds) – we do, however, believe that many of the target applications do not have such strict latency constraints.

5. RELATED WORK

Efforts to decrease the power consumption of wireless devices were made at all networking layers. At the physical layer, efforts have included utilizing lower-power radio propagation techniques, such as code division spread spectrum (e.g., DSSS and FHSS [35]). At the datalink layer, medium access control (MAC) protocols govern when and how the nodes of the network coordinate to share a broadcast channel effectively. CSMA MAC protocols [6, 24, 11] are generally wasteful inasmuch as nodes must constantly monitor the broadcast channel in Rx mode – an expensive radio function. Other protocols, such as TDMA [20, 23], have been designed to allow nodes to power down their antennas during particular time-slots of a predetermined schedule. However, these work do not consider network topology and application-specific data interests.

Imielinski's work [17], where a wireless server broadcasts data items according to a temporal *directory*, also has the goal of reducing power consumption. The directory tells clients when to listen for particular data items. This is analagous to the downstream control event in TD-DES. Another related model is that of *broadcast disks* [7] proposed for asymmetric systems in which a server broadcasts a rotating schedule, or *disk*, of data downstream to clients. This model was later extensively studied and extended for hierarchical data dissemination (e.g., [15]). TD-DES borrows from both models the basic notion of disseminating events through a schedule. TD-DES also assumes a multi-hop network environment where the topology of the network can be arbitrary and needs to be taken into account, which significantly complicates the problem. Furthermore, TD-DES is designed for symmetric data flow, downstream as well as upstream, whereas the other protocols address asymmetric, downstream communication only.

Recent work on wireless publish/subscribe [16] also uses soft-state trees for data dissemination. This work describes several heuristics for dissemination tree construction but does not address power-management through scheduled power-down periods.

Directed diffusion [19], TAG [26], and the COUGAR [37] address data-centric extraction of information from wireless sensor networks, commonly through intelligent in-network data aggregation. Our approach can be regarded as a generalization of TAG's *epoch-based* aggregation approach. In the TAG approach, aggregation takes place over time as a sequence of epochs, each consisting of data propagation from children to parent nodes and generic data aggregation at the parents. The TAG approach is geared specifically towards upstream aggregation, whereas our approach is a step towards generic upstream and downstream event-based data dissemination. Furthermore, the scheduling issues we investigate have not been addressed by prior work.

Most relevant to our approach are topology management schemes that address the issue of which nodes should turn their radios off and when. S-MAC [38] allows nodes to go to sleep periodically. Nodes go to sleep for some time, and then wake up to listen to see if other nodes want to talk to them. Sleep schedules can be adjusted on a per-node basis. In STEM [34], nodes use two seperate radios, one acting as a low-power paging channel and one as a data transmission channel, to trade off power savings for path set up latency. Nodes use the paging radio to detect tranmission requests from their neighbors, and then wake up their primary radio to pick up the actual data transmission. TD-DES shares similar goals with these topology management schemes. Unlike these schemes, however, TD-DES also addresses application semantics-based data dissemination and scheduling.

6. CONCLUSIONS AND FUTURE WORK

We described TD-DES, an integrated scheduling and data dissemination model and protocol for multi-hop networks of energy-constrained devices such as sensor networks. The model is based on event-based communication, one that integrates the division of the shared wireless medium access based on *network topology* with the *application specific semantics* of the data to be disseminated.

The network is arranged as an event dissemination tree, with nodes subscribing to the event types they are interested in. The event scheduler dynamically allocates and multiplexes upstream and downstream time slots for each event type. Power consumption among wireless nodes is reduced by allowing each node to listen to the wireless channel during the transmission times of data items which *interest* them (or their topological dependents)– at other times, they are allowed to save energy by powering down their radios.

We also presented preliminary performance results, based on a detailed simulation model, that demonstrate significant potential power savings of the proposed approach. We argue that while TD-DES is efficient from a power consumption perspective, it clearly suffers generally from worse multi-hop dissemination latencies for generated events than does its non-scheduled counterpart network (which provides optimal latency at the expense of maximal power consumption).

In addition to conducting a more comprehensive quantitative study of the proposed approaches, we plan to address the following issues as part of our future work:

Implementation and Clock Synchronization. We are currently implementing our protocols on top of Mica sensors (a.k.a. motes) [1] running TinyOS [5]. A key implementation issue that needs be addressed is *clock synchronization*. TD-DES calls for each child to have its clock internally synchronized with that of its parent for the protocols to work correctly. The protocols require that the clock errors are insignificant relative to the duration of individual time slots. While traditional approaches designed for wired networks (e.g., NTP [18]) may not provide sufficiently high precision, recent proposals that were designed specifically for wireless sensor networks [9, 8] lower the error rates down to several microseconds. Such a precision is good enough for our purposes, because we expect a typical slot duration to be in the order of milliseconds.

Mobility and Reliability. In this initial study, we have focused only on static topologies. The dissemination tree construction protocol allows dynamic topology changes and can therefore cope with node mobility. On the other hand, situations may arise when packets may not arrive to their destinations during node movements. Reliable delivery can be ensured by using a straightforward ack-retransmit scheme integrated with caching: if a node received an event with a sequence number, but had not received the previous sequence number, it might request its parent to resend the event. The challenge here is to ensure reliability in such a way that the power efficiency of the system is not compromised.

Another related reliability concern arises due to the use of a tree-based dissemination topology. Even though the tree structure is self organizing and can adapt quickly to node failures and disconnections, mesh-based organizations [28] are inherently more reliable as they provide multiple paths between network nodes. Extending the protocols to mesh topologies is an interesting research direction.

Caching and Aggregation. Recent work has focused on upstream aggregation techniques [26, 18] to reduce the volume of data transmission. TD-DES can support upstream aggregation as well as much less studied *downstream* aggregation where the key idea is to aggregate multiple events of the same type at the root before propagating them downstream.

7. REFERENCES

[1] Crossbow Mica Motes. http://www.xbow.com.

[2] Great duck island homepage. http://www.greatduckisland.net/.

[3] IEEE 802.11. http://standards.ieee.org/getieee802/802.11.html.

[4] The official Bluetooth website. http://www.bluetooth.com/.

[5] The TINYOS Web site. http://www.cs.berkeley.edu/ jhill/tos/.

[6] P 802.11; draft wireless LAN medium access control (MAC) and physical layer (PHY) specifications. Technical report, New York, 1997.

[7] S. Acharya, R. Alonso, M. J. Franklin, and S. B. Zdonik. Broadcast disks: Data management for asymmetric communications environments. In *Proceedings of the 1995 ACM SIGMOD*, pages 199–210, San Jose, California, 22–25 May 1995.

[8] J. Elson and D. Estrin. Time synchronization for wireless sensor networks. In *Proceedings of the 15th International Parallel and Distributed Processing Symposium (IPDPS-01)*, pages 186–186, Los Alamitos, CA, Apr. 23–27 2001.

[9] J. Elson, L. Girod, and D. Estrin. Fine-grained network time synchronization using reference broadcasts. In *5th Symposium on Operating Systems Design and Implementation (OSDI'02)*, Boston, MA, Dec 2002.

[10] D. Estrin, L. Girod, G. Pottie, and M. Srivastava. Instrumenting the world with wireless sensor networks. In *Proceedings of ICASSP 2001*, Salt Lake City, Utah, May 2001.

[11] C. L. Fullmer and J. J. Garcia-Luna-Aceves. FAMA-PJ: a channel access protocol for wireless LANs. *Proceedings of the Annual International Conference on Mobile Computing and Networking, MOBICOM*, pages 76–85, 1995.

[12] R. Govindan, T. Faber, J. Heidemann, and D. Estrin. Ad hoc smart environments. In *Proceedings of the DARPA/NIST Workshop on Smart Environments*, Atlanta, June 1999.

[13] G.Pottie and W. Kaiser. Wireless integrated network sensors. *Communications of the ACM*, 3(5):51–58, May 2000.

[14] J. Haartsen, M. Naghshineh, J. Inouye, O. Joeressen, and W. Allen. Bluetooth: Vision, goals, and architecture. *Mobile Computing and Communications Review*, 5(6):784–803, December 1998.

[15] Q. Hu, D. L. Lee, and W.-C. Lee. Performance evaluation of a wireless hierarchical data dissemination system. In *Mobile Computing and Networking*, pages 163–173, 1999.

[16] Y. Huang and H. Garcia-Molina. Publish/subscribe tree construction in wireless ad-hoc networks. Technical report, Stanford University, Department of Computer Science, 2001.

[17] T. Imielinski, S. Viswanathan, and B. R. Badrinath. Energy efficient indexing on air. In *ACM SIGMOD*, pages 25–36, 1994.

[18] C. Intanagonwiwat, D. Estrin, R. Govindan, and J. Heidemann. Impact of network density on data aggregation in wireless sensor networks. In *Proceedings of International Conference on Distributed Computing Systems (ICDCS)*, Vienna, Austria, July 2002.

[19] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *Proceedings of the International Conference on Mobile Computing and Networks (MOBICOM)*, Boston, Massachusetts, August 2000.

[20] C. E. Jones, K. M. Sivalingam, P. Agrawal, and J.-C. Chen. A survey of energy efficient network protocols for wireless networks. Technical report, School of EECS, Washington State University, Pullman, WA 99164, 2001.

[21] D. Jonhson and D. Maltz. *Mobile Computing*, chapter 5 (Dynamic Source Routing). Kluwer Academic Publishers, 1996.

[22] J. Kahn, R. Katz, and K. Pister. Next century challenges: mobile networking for smart dust. In *International Conference on Mobile Computing and Networking (MOBICOM '99)*, pages 271–278, Aug. 1999.

[23] J. Kardach. Bluetooth architecture overview. *Intel Technology Journal*, (Q2):7, May 2000.

[24] P. Karn. Maca – a new channel access method for packet radio. In *ARRL/CRRL Amateur Radio 9th Computer Networking Conference*, pages 134–40, 1990.

[25] P. Levis and D. Culler. Mate: A tiny virtual machine for sensor networks. In *Proceedings of the Tenth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, San Jose, CA, October 2002.

[26] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. Tag: A tiny aggregation service for ad hoc sensor networks. In *Proceedings of the OSDI Conference*, December 2002.

[27] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. The design of an acquisitional query processor for sensor networks. In *ACM SIGMOD*, San Diego, June 2003.

[28] E. L. Madruga and J. Garcia-Luna-Aceves. Multicasting along meshes in ad-hoc networks. In *Proceedings of the IEEE International Communications Conference (ICC)*, June 1999.

[29] C. E. Perkins and P. Bhagwat. Highly-dynamic destination-sequenced distance vecor routing (DSDV) for mobile computers. In *Proceedings of the ACM SIGCOMM'94 Conference*, pages 234–244, August 1994.

[30] C. E. Perkins and E. M. Royer. Ad-hoc on-demand distance vector (AODV) routing. In *Proceedings of the 2nd annual IEEE Workshop on Mobile Computing Systems and Applications*, February 1999.

[31] P. Saas. Communications networks for the force XXI digitized battlefield. *ACM/Baltzer Mobile Networks and Applications Journal*, October 1999.

[32] P. Saffo. Sensors: The next wave of information. *Communications of the ACM*, 40(2):92–97, February 1997.

[33] S. Schlosser, J. Griffin, D. Nagle, and G. Ganger. Designing computer systems with mems-based storage. In *Proceedings of ASPLOS*, June 2000.

[34] C. Schurgers, S. Ganeriwal, and M. Srivastava. Optimizing sensor networks in the energy-latency-density design space. *IEEE Transactions of Mobile Computing*, 1(1), Jan 2002.

[35] M. K. Simon, J. K. Omura, S. R. A., and B. K. Levitt. *Spread Spectrum Communications Handbook*. McGraw-Hill, New York, NY, USA, 1994. ISBN 0-07-057629-7.

[36] C.-K. Toh. *Ad Hoc Mobile Wireless Networks: Protocols and Systems*. Prentice-Hall, Englewood Cliffs, NJ 07632, USA, 2002.

[37] Y. Yao and J. E. Gehrke. Query processing in sensor networks. In *First Biennial Conference on Innovative Data Systems Research (CIDR 2003)*, Asilomar, California, January 2003.

[38] W. Ye, J. Heidemann, and D. Estrin. An energy-efficient MAC protocol for wireless sensor networks. In *IEEE Infocom*, June 2002.