

# Efficient Power Management based on Application Timing Semantics for Wireless Sensor Networks

Octav Chipara, Chenyang Lu, and Gruia-Catalin Roman  
Department of Computer Science and Engineering  
Washington University in Saint Louis

## Abstract

*This paper proposes Efficient Sleep Scheduling based on Application Timing (ESSAT), a novel power management scheme that aggressively exploits the timing semantics of wireless sensor network applications. We present three ESSAT protocols each of which integrates (1) a lightweight traffic shaper that actively shapes the workload inside the network to achieve predictable timing properties over multiple hops, and (2) a local scheduling algorithm that wakes up nodes just-in-time based on the timing properties of shaped workloads. Our ESSAT protocols have several distinguishing features. First, they can save significant energy with minimal delay penalties. Second, they do not maintain TDMA schedules or communication backbones; as such, they are highly efficient and suitable for resource constrained sensor platforms. Moreover, the protocols are robust in highly dynamic network environments, i.e., they can handle variable multi-hop communication delays and aggregate workloads involving multiple queries, and can adapt to varying workload and network topologies. Our simulations showed that DTS-SS, an ESSAT protocol, achieved an average node duty cycle 38-87% lower than SPAN, and query latencies 36-98% lower than PSM and SYNC.*

## 1 Introduction

Energy is the most critical resource in wireless sensor networks (WSNs) that must operate for years on limited power supplies. Recent studies have shown that significant energy savings can be achieved by dynamically managing node duty cycles. However, the design of power management protocols faces several key challenges. First, the network must maintain sufficient quality of service despite sleep schedules. In particular, many mission-critical applications operate under stringent timing constraints. For example, a surveillance application may require the network to report all suspicious events within a few seconds in order to insure timely response

to intrusions. Power management protocols designed for such applications must coordinate the sleep schedules of different nodes to minimize their impact on end-to-end communication delays. Second, hardware platforms in WSNs usually have limited bandwidth, memory, and processing capabilities. A practical power management protocol must be simple and introduce minimal overhead. Third, the workload in WSNs may change dramatically in response to events in the physical environment. For instance, while the workload in a fire monitoring system may be moderate during normal conditions, it may increase sharply after a wild fire is detected to support numerous fire fighting activities. Therefore, a power management protocol must be able to dynamically adjust the duty cycles of nodes based on the current system workload. Furthermore, a power management protocol must be robust against node and link failures, which can occur frequently in WSNs [17].

We present *Efficient Sleep Scheduling based on Application Timing (ESSAT)*, a novel power management scheme that meets all the above challenges by aggressively exploiting *application timing semantics*. In contrast to general-purpose systems with random workloads, workloads in WSNs are often generated by applications with known timing semantics. A primary function of many WSN applications is to continuously gather data from the environment at user-specified periods. Moreover, in many distributed signal processing applications (e.g., target detection), multiple sensor nodes sample and exchange data at application-specific sampling frequencies for data fusion. Intuitively, a power management protocol can leverage such application-level timing semantics in order to optimize sleep schedules. However, the design of such protocols is complicated by several issues. The random back-off scheme in widely adopted CSMA/CA MAC protocols can cause variable communication delays due to channel contention. More importantly, the delay jitter can accumulate over multiple hops. As a result, even when data is generated periodically at sources, the workload often becomes highly *aperiodic* over multi-hop communication.

In addition, the *aggregate* network workload of multiple periodic data flows may be *aperiodic* due to their different periods and starting times.

ESSAT deals with the above complexities with two efficient mechanisms: (1) *in-network traffic shapers* that actively control packet transmission to preserve predictable timing properties inside the network; and (2) a local sleep scheduler called *Safe Sleep* that wakes up nodes *just in time* to meet communication needs. ESSAT is optimized for WSNs. It saves significant energy while introducing minimal delay penalties. Our ESSAT protocols also address a number of important practical issues in WSNs. In contrast to many existing protocols (discussed in Section 2), our protocols do not maintain TDMA schedules or active communication backbones; therefore, they are highly efficient and suitable for resource constrained sensor platforms. Moreover, our protocols are robust in highly dynamic wireless sensor networks, i.e., they can adapt to varying workloads induced by multiple queries as well as node failures.

The remainder of the paper is organized as follows. In Section 2, we highlight the contributions of ESSAT by contrasting it with related work. A workload model with application timing semantics is formulated in Section 3. Section 4 presents the design and analysis of Safe Sleep and two traffic shaper algorithms. Experimental evidence regarding the effectiveness of the proposed protocols is discussed in Section 5. Concluding remarks appear in Section 6.

## 2 Related Work

ESSAT is based on two high-level design principles: (1) shaping the traffic inside a network to achieve predictable timing properties, and (2) exploiting application-level timing semantics. In the following, we discuss existing power management schemes and the extent to which they explore similar principles.

An approach adopted by several power management protocols is to maintain a connected communication backbone that is responsible for routing packets, while other nodes sleep most of the time [3, 15, 13]. Although the backbone provides good communication performance by maintaining sufficient network connectivity, this solution does not exploit the possibility of conserving energy on the backbone nodes even when they are not needed for communication. Keeping the backbone nodes continuously active may become unacceptable in WSNs especially when the workload is light. Power management schemes based on communication backbones do not exploit traffic shaping or workload characteristics.

The idea of using traffic shaping to facilitate sleep scheduling has been explored in power management schemes that operate at the MAC layer [9, 4, 16, 12, 11,

18]. Traffic shaping may be performed at different levels of granularity. TDMA MAC protocols [9, 4] perform fine-grained traffic shaping by allocating time slots to each node. A node only communicates in its slots, and can sleep in the others. However, maintaining fine-grained TDMA schedules for a multi-hop sensor network is challenging. Centralized scheduling algorithms cannot scale effectively in large networks while distributed scheduling algorithms can introduce significant synchronization overhead in order to maintain consistent schedules in multi-hop networks [9]. A simpler scheme for traffic shaping is based on coarse-grained sleep schedules [16, 1, 11, 18] in which each node follows a fixed periodic schedule that includes an active window and a sleeping window. However, none of the above traffic shaping schemes consider the workload characteristics when constructing sleep schedules. Consequently, they may introduce significant delay penalties when their schedules interfere with the timing semantics of the application workload.

Several power management schemes exploit workload properties available at different layers to improve their energy efficiency. For example, T-MAC [12] and PSM [1] adapt a node's duty cycle in response to the network load observed at the MAC layer, while on-demand power management [19] uses routing information. However, neither solution is cognizant of the timing semantics at the application layer. As a result, they may introduce delay penalties or waste energy due to the lack of precise timing information of the workload.

TinyDB [8] allows a user to collect aggregated data from a sensor network through a routing tree. It evenly divides the period of a query into communication slots for nodes at different levels in the routing tree, and nodes can sleep in slots assigned to other levels. TinyDB does not address sleep scheduling for multiple queries with different timing properties. Moreover, the duty cycle of each node is fixed and does not adapt to the workload.

In contrast to the aforementioned approaches, ESSAT employs a novel power management approach based on light-weight traffic shaping and the timing semantics of WSN applications. The traffic shaping algorithms introduce minimal delay penalties and communication overhead. Furthermore, they can efficiently adapt to the dynamics in the network and aggregate workload of multiple queries. This unique combination of features makes ESSAT especially suitable for real-time applications on resource-constrained WSNs.

## 3 Workload Model

In this paper we assume a general workload model in which each source produces data reports periodically for a query. This model fits many WSN applications that gather data from the environment at user specified rates.

Such applications generally rely on existing WSN query services. We analyze the ESSAT protocols on the basis of the workloads produced by a generic query service.

A query is characterized by the following parameters: a set of sources that respond to registered queries, an aggregation function [8] for in-network aggregation; the period  $P$  at which data reports are generated by the sources; and the starting time of the query  $\phi$ .

A query service usually works as follows: a user issues a query to a sensor network through a base station, which disseminates the query to all the sources. To facilitate data aggregation, the query service constructs a *routing tree* rooted at the base station as the query is disseminated. During the execution of the query, each leaf node generates a new data report every  $P$  seconds. The first data report is generated by the leaf nodes at time  $\phi$ . Each non-leaf node waits to receive the data reports from its children, produces a new data report by aggregating its data with the children’s data reports, and then sends it to its parent.<sup>1</sup>

Although in our model the sources produce data reports periodically for query we do not assume that the network workload remains periodic. In fact, as discussed in Section 5.3 the workload is aperiodic due to multi-hop delay jitter and multiple queries with different timing properties. Since this query model approximates several representative query services [5, 8], ESSAT can be easily integrated with existing query services to support various WSN applications. ESSAT can also be extended to support other communication patterns such as peer-to-peer communication or data dissemination.

## 4 Protocols Design

An ESSAT protocol has two components: a traffic shaper and a sleep scheduling algorithm called Safe Sleep. The traffic shaper controls the sending and receiving of data reports to construct workloads with predictable temporal properties. Based on the temporal properties of the workload, Safe Sleep determines when a node should be turned on or off.

The section starts by introducing Safe Sleep. We then consider the behavior of the WSNs in the absence of traffic shaping. Next, we introduce two traffic shapers that improve the energy efficiency. Finally, we analyze the behavior of the ESSAT protocols in the presence of packet drops and topology changes.

ESSAT is layered between the MAC protocol and the query service. ESSAT does not require special scheduling support at the MAC layer. For example, it can work with 802.11b and the CSMA/CA protocol of TinyOS [14].

<sup>1</sup>To deal with node failures, a node may timeout and send its data report to its parent before receiving the data reports from all its children (see Section 4.1).

### 4.1 Safe Sleep (SS)

Safe Sleep (SS) is a local sleep scheduling algorithm that turns the radio on and off. From the point of view of SS, a node may be in one of two states: *free* or *busy*. A node is *busy* when it expects to receive or send a data report. Otherwise, a node is *free*. SS determines a node’s state based on the timing properties of its workload and schedules periods of sleep and activity accordingly.

**Timing properties of queries.** The timing properties of a node’s query workload are shared by SS and the traffic shapers. A node characterizes each query as follows. Since the nodes are organized in a tree topology, a node expects a data report from each child in each query period. Let  $r(q, k, c)$  be the *expected reception time* of the  $k^{\text{th}}$  data report for query  $q$  from child  $c$ . The *expected send time* of the  $k^{\text{th}}$  data report for query  $q$ ,  $s(q, k)$ , is the time when the data report is scheduled to be submitted to the MAC layer for transmission. SS keeps track of all queries routed through a node. For each query  $q$ , the node stores the time it expects the next data report from each child in  $q.r_{next}(c)$  and the time it expects to send the next aggregated data report to its parent in  $q.s_{next}$ . It is the responsibility of the traffic shaper to control the times when the next data reports are to be received or sent. This information is provided to SS in an incremental manner by the traffic shaper as follows. Upon receiving a data report for query  $q$  from child  $c$ , the traffic shaping protocol computes  $r(q, c, k + 1)$  while upon completing the sending of a data report the traffic shaper computes  $s(q, k + 1)$ . The traffic shapers are presented in Section 4.2.

**Algorithm.** SS works as follows. A node checks its state after it sends or receives a data report. Let  $t_{wakeUp}$  be the minimum of the expected reception and send times of all queries. If  $t_{wakeUp}$  is larger than the current time, then the node remains free until  $t_{wakeUp}$ . Otherwise the node is busy since data reports are to be received or sent. If a node is free, SS may turn off its radio. However, to avoid incurring any delay or energy penalties the costs associated with transitioning between power states must be considered. To characterize these costs we define the break-even time  $t_{BE}$  as the minimum time the node needs to remain free such that there is no delay or energy penalty in turning the radio off and back on [2]. When the radio’s transition power is no higher than its active power, the break-even time is the time it takes to transition from the active state to the off state ( $t_{ON \rightarrow OFF}$ ) and back ( $t_{OFF \rightarrow ON}$ ). A method for computing the break-even time when the transition power is higher than the active time is given in [2]. SS ensures that no energy or delay penalties are incurred through two steps: First, SS puts the node to sleep only if the nodes is free and remains free for longer than the break-even time. Second,

```

updateNextReceive( $q, c, r(q, k + 1, c)$ ) {
  Update the next expected receive time
   $q.r_{next}(c)$  with  $r(q, k + 1, c)$ ;
  checkState(); }
updateNextSend( $q, s(q, k + 1)$ ) {
  Update the next expected send time
   $q.s_{next}$  with  $s(q, k + 1)$ ;
  checkState(); }
checkState() {
   $t_{wakeUp} = \min(\{t | t = q.s_{next} \ \forall q, c\}$ 
     $\cup \{t | t = q.r_{next}(c) \ \forall q\})$ 
   $t_{sleep} = t_{wakeUp} - now$ ;
  if ( $t_{sleep} > t_{BE}$ )
    sleep and set time to wake up at
    ( $t_{sleep} - t_{OFF \rightarrow ON}$ ); }

```

**Figure 1.** Safe Sleep Algorithm

the node sleeps until  $t_{sleep} - t_{OFF \rightarrow ON}$  such that there is enough time to wake up. We call this algorithm Safe Sleep because it guarantees that no energy or delay penalties are incurred by turning the node off.

So far we have considered the operation of the system after the query setup is performed. The root starts a query by flooding a query request to all sources. During the setup slot, all nodes keep their radio on even if SS does not expect any data reports to be sent or received. For the duration of the setup slot both setup requests and data reports may be transmitted. Outside the setup slot, only data reports may be transmitted. The size of the setup slot affects both the time it takes to setup a query and the energy consumption at a node.

We note that SS has two notable features. First, it can flexibly schedule sleep periods for multiple queries. It does not need to maintain a TDMA schedule. Second, the storage cost of each query is proportional to the degree of the node in the routing tree. This localized property allows SS to scale effectively in large networks.

**Impact of incorrect predictions.** When the prediction of the workload is perfect, i.e., the *expected* reception time  $r(q, k, c)$  and the *actual* reception times coincide, SS achieves the maximum sleep time. However, as discussed earlier, it is difficult to predict the actual reception time in WSNs due to delay jitter.

The accuracy of the expected reception time greatly affects the ability of SS to conserve energy. When the actual reception time is earlier than the expected reception time, a data report cannot be transmitted successfully because the receiver may be asleep. To avoid transmission failures, the traffic shapers always set the expected reception time of a child's data report to be the same as the child's expected send time of the same data report. Even if a data report is ready before its expected send time, the sender buffers it until the expected send time when the receiver wakes up.

Inaccuracies in the expected reception time lead to

shorter sleep intervals because SS keeps the radio turned on from the time the data report is expected until the data report arrives. This situation has two possible causes. First, a child may not be able to send its data report to the MAC layer at the expected send time because it has not received the data reports from its own children due to communication delays. Second, even after a child sends the data report to its MAC layer, the data report suffers from additional delays due to queuing, contention, and transmission at the MAC layer.

## 4.2 Traffic shapers

In this subsection, we first analyze the performance of SS without traffic shaping and then present two traffic shapers that improve the energy efficiency with minimal delay penalty.

### 4.2.1 No Traffic Shaping (NTS)

In the following we describe how SS can be used without traffic shaping. In this case, SS only takes advantage of the periodicity of the data reports from the leaves of the routing tree. Since each node performs aggregation, a non-leaf node usually waits to receive data reports from its children before it transmits the aggregated data report. In NTS, a node sends its aggregated data reports to its parent *immediately* after it has received and aggregated the data reports from its children.

As mentioned earlier, even though data reports are produced periodically on each leaf node, the time when a node actually receives a data report is unpredictable due to accumulated jitter in multi-hop wireless communication. NTS estimates the reception times of data reports as follows. The reception time of the  $k^{th}$  data report of query  $q$  must be no earlier than the beginning of a query period  $\phi + k * P$ , where  $\phi$  and  $P$  are the start time and the period of the query, respectively.<sup>2</sup> Every node shares the same expected send and reception times of the  $k^{th}$  data report:  $s(k) = r(k) = \phi + k * P$ . For a single query, a node turns on its radio at  $r(k)$  and keeps listening until it receives all data reports from its children, performs the aggregation, and relays the aggregated data report to its parent. After the node sends the data report, it can turn off its radio until  $r(k + 1)$  when a new data report is produced. By design, SS also handles concurrent queries.

The protocol that employs SS without traffic shaping is denoted by NTS-SS. An advantage of NTS-SS is that it does not introduce any delay penalties. However, NTS-SS wastes energy by pessimistically turning on all nodes when a data report is generated on the leaf nodes.

<sup>2</sup>Previously the notation  $r(q, k, c)$  was used to refer to the reception time of the  $k^{th}$  data report of query  $q$  from child  $c$ . When no ambiguity exists we will drop the variables  $q$  and  $c$  for brevity.

**Analysis.** We analyze the performance of NTS-SS in terms of energy efficiency and query latency. For clarity we only consider the case when a single query is registered. To evaluate the energy efficiency of our protocols we quantify the time a node is active. A node may be active because it expects data reports from its children or because it is sending data reports. The active time for sending a data report is not directly affected by the traffic shaper, making this case uninteresting for power analysis. Thus, we focus on analyzing the time a node remains awake to receive data reports from its children  $T_{recv}$ .

The energy conserved by a node using NTS-SS varies as a function of its *rank* in the tree. We define the rank  $d$  of a node to be the maximum hop count to any of its descendants in the routing tree. By definition a leaf node has a rank of zero.

For a leaf node,  $T_{recv}$  is zero since it does not receive data reports. The time a node with rank  $d > 0$  remains active is the sum of three components: the maximum time it takes for each child to receive the data report  $T_{recv}(d-1)$ ; the maximum time it takes its children to compute their data reports by applying the aggregate function  $T_{comp}$ ; and the maximum time it takes to receive all data reports from its children  $T_{collect}$ . Therefore,  $T_{recv}(d) = T_{recv}(d-1) + T_{collect} + T_{comp}$ . Let  $T_{agg}$  be the upper bound on the time it takes for a node to receive all the data reports from all its children and generate a data report:  $T_{agg} = T_{collect} + T_{comp}$ . Accordingly,  $T_{recv}(d)$  is:

$$T_{recv}(d) = \begin{cases} 0, & \text{if } d = 0; \\ (d-1) * T_{agg} + T_{collect}, & \text{if } d \neq 0. \end{cases} \quad (1)$$

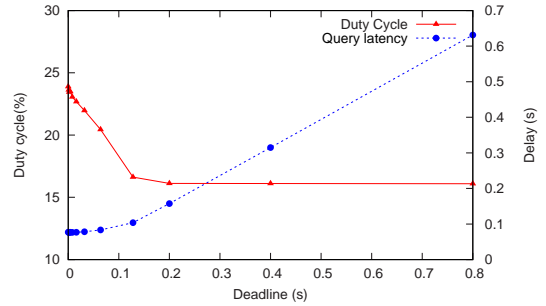
Note that large variations in the energy conserved at different nodes limits the lifetime of the network. The nodes close to the root that have higher ranks will run out of energy faster than the others. Therefore, this NTS-SS exhibits good energy efficiency only when the routing tree is small.

We observe that the energy efficiency of NTS-SS may be improved by increasing the control over when the data reports are generated. This improves the accuracy of expected reception times at intermediary nodes, thus reducing the idle listening time. To this end, we introduce two traffic shapers.

#### 4.2.2 Static Traffic Shaper (STS)

STS enforces the periodicity of data reports by pacing their multi-hop transmission over a period equal to an assigned *deadline*  $D$ . In our implementation, we allocate the same amount of time  $l$  to each rank in the tree. We let the local deadline  $l$  be:  $l = \frac{D}{M}$  where  $M$  is the maximum rank of the tree.

For each query, the expected reception time of a node of rank  $d$  is  $r(k) = \phi + k * P + l * (d-1)$  and its expected



**Figure 2.** Impact of query deadline on duty cycle and query latency of STS-SS.

send time is  $s(k) = \phi + k * P + l * d$ . If a data report is generated before its expected send time  $s(k)$  it is buffered until that time. If the data report is late, then the node sends it immediately. This mechanism reduces the difference between the expected and actual send and reception times and hence improves the energy efficiency of STS.

**Analysis.** Let STS-SS denote the ESSAT protocol that integrates STS and SS. We consider the case when a single query runs in the system. The critical parameter that must be tuned in STS is the local deadline  $l$ . The choice of  $l$  represents a tradeoff between the energy efficiency and query latency. We now analyze the impact of  $l$  on STS's query latency and the energy efficiency. Query latency is the maximum time it takes to send a data report from any source to the root. For STS, the query latency is:

$$L_q = M * \max(l, T_{agg}) \quad (2)$$

When  $l < T_{agg}$  a node with rank  $d$  expects to receive the data reports from its children at  $l * (d-1)$  and turns on its radio at that time. However, since  $l < T_{agg}$  the children cannot send the data reports on time. The data reports reach a node with rank  $d$  at  $T_{agg} * (d-1) - T_{collect}$ . Thus,  $T_{recv} = T_{agg} * (d-1) - T_{collect} - l * (d-1) = (T_{agg} - l) * (d-1) + T_{collect}$ . When  $l \geq T_{agg}$  the time the node remains awake to receive the data reports is  $T_{collect}$  since the children are ready to transmit their data reports in time. Therefore,

$$T_{recv}(l, d) = \begin{cases} 0, & \text{if } d = 0; \\ (T_{agg} - l) * (d-1) + T_{collect}, & \text{if } l \leq T_{agg} \wedge d \neq 0; \\ T_{collect}, & \text{if } l > T_{agg} \wedge d \neq 0. \end{cases} \quad (3)$$

Note that in the special case when  $l = 0$ , STS behaves like NTS.

Figure 2 shows the impact of the query deadline on the average duty cycle and query latency obtained in an *ns-2* simulation when three queries are running. The complete experimental setup is described in Section 5. The duty cycle of a node is defined as the percentage of time a node remains active during a query. A discontinuity point is observed in both average duty cycle and query latency at  $D = 0.12s$  when the local deadline

$l$  approaches  $T_{agg}$ . When  $D < 0.12s$ , the query latency remains almost constant, while the average node duty cycle decreases monotonically as  $D$  increases. On the other hand, when  $D > 0.12s$ , the query latency increases proportionally with the deadline without reducing the average duty cycle. This result validates the above analysis.

STS-SS has maximum energy efficiency and good query latency when its local query deadline  $l$  approaches  $T_{agg}$ . However, due to the dynamic nature of WSNs it is difficult to estimate  $T_{agg}$  accurately. Hence, it is difficult to tune the parameter  $l$  of STS. To address this limitation we develop the Dynamic Traffic Shaper.

### 4.2.3 Dynamic Traffic Shaper (DTS)

In contrast to STS which assigns fixed expected reception and send times, DTS dynamically adapts the expected send and reception times in response to variations in the multi-hop delays of received data reports.

Similarly to NTS, DTS initially sets the expected send and reception times to equal the start time of the query:  $s(0) = r(0) = \phi$ . Every time a node sends a new data report, DTS sets the expected send time of its next data report to  $s(k) = s(k-1) + P$ . Depending on whether or not the  $k^{th}$  data report is sent at the expected send time, DTS behaves differently. If a node receives the data reports from all of its children in time such that its  $k^{th}$  data report is ready before  $s(k)$ , it sends it at  $s(k)$  and computes the next expected send time as  $s(k+1) = s(k) + P$ . After receiving the data report, the parent sets its next expected reception time to  $r(k+1) = r(k) + P$ . No explicit synchronization between a node and its parent is necessary in this case. However, when the  $k^{th}$  data report is ready at a time  $t > s(k)$ , DTS sends the data report immediately and sets the next expected send time  $s(k+1) = t + P$ . This case is called a *phase shift*. When a phase shift occurs, the node piggybacks  $s(k+1)$  in the packet containing the data report. After receiving the data report, the parent sets its next expected reception time to  $s(k+1)$ .

We note that, the expected reception and send times are computed similarly to the Release Guard protocol[10] developed for multiprocessor real-time systems. However, Release Guard and DTS are designed for different purposes and systems. Release Guard is used to compute task release times to allow end-to-end schedulability analysis for real-time systems. In contrast, DTS is used as a traffic shaper for efficient sleep scheduling in WSNs. Unlike Release Guard that deals with chains of tasks, DTS handles data aggregation on multi-hop routing trees. Moreover, to improve energy efficiency, DTS re-synchronizes a child and its parent through an explicit packet exchange when data reports are dropped (discussed in Section 4.3) while Release Guard does not

support this mechanism because it is not concerned with sleeping nodes.

**Analysis.** As described above, if a node does not receive a child's data report by the expected reception time, it wastes energy listening until the report arrives. To prevent future energy wastage, the node performs a phase shift to postpone its expected send time of the next data report from that child. Essentially, DTS *adapts* to the network workload by adjusting the expected send and reception times of data reports based on the longest multi-hop delay of received data reports.

The adaptive feature of DTS is accomplished at the cost of additional synchronization overhead. However, since DTS advertises the expected send time of the next data report *only* when a phase shift occurs its communication overhead is small. To verify this conjecture, we run DTS in the presence of three queries with different rates under the experimental setup described in Section 5. On average the overhead due to piggybacked phase updates is less than *one bit per data report* for all tested query rates. (The detailed results are not shown due to space limitations). The low overhead indicates that DTS is suitable for bandwidth-constrained WSNs.

Through protocol analysis we observed a tradeoff between energy conservation and query latency. Transmitting the data reports immediately, as in the case of NTS, does not incur any delay penalties, but this comes at the cost of poor energy efficiency. In contrast, STS and DTS conserve additional energy through traffic shaping at the cost of slightly increased delay. In STS, the parameter  $l$  controls the tradeoff between energy efficiency and query latency. Since tuning  $l$  is difficult, DTS has the practical advantage of being self-tuning.

### 4.3 Protocol Maintenance

A key feature of the ESSAT protocols is their robustness in face of network dynamics. We are interested in analyzing the behavior of the ESSAT protocols under two failure modes: when packets are transiently lost and when the topology changes due to persistent link and node failures.

**Transient packet loss.** In the case of transient packet loss, NTS-SS and STS-SS require no corrective action because the expected reception and send times are independent of the node's parent or child. In contrast, in DTS-SS, when a data report that contains a phase update is dropped the nodes become unsynchronized. When a packet loss is detected (e.g., based on the sequence numbers of received data reports), a node resynchronizes its schedule as follows. If the data report received after the transient packet drop(s) contains a phase update, this phase is used as the new phase for DTS-SS. Otherwise, the receiver requests a phase update from the sender. If

the transmission of a data report is acknowledged, the receiver may piggyback the request for a phase update in the acknowledgement packet. Otherwise, a new packet is sent to request a phase update. The sender then piggybacks the expected send time in the next data report. Because a phase shift *only delays* the expected send time of future data reports, the receiver can tolerate the loss of multiple consecutive phase updates. However, this leads to transient energy waste because the node remains awake until the sleep schedules are resynchronized.

**Topology changes.** In the case of a persistent node or link failures the query service or routing protocol is responsible for reconfiguring the routing tree. When a node fails both the parent and the children of the failed node need to recover from the failure.

A node discovers that it is the parent of a failed node if one of its children repeatedly fails to deliver its data report. In this case, all ESSAT protocols take two actions. First, the parent removes its dependency on the failed node, such that it no longer waits for data reports from a failed child. Second, the stale expected send and reception times of the failed node used by SS are removed.

A node discovers that it is the child of a failed node if it repeatedly fails to transmit its data report to its parent. The first step in the recovery process is for the query service or routing protocol to identify a new parent. The new parent adds a dependency on the node such that it generates aggregated data report after the child contributed its data report. The second step is to update (if necessary) the expected send and reception times of the node. NTS-SS does not require an update since all nodes share the expected send and reception times. In contrast, for STS-SS,  $s(k)$  and  $r(k)$  depend on the rank of the node. As such, when the parent is changed the rank of the node may also change. When the rank changes, the considered node and its descendants must recompute  $s(k)$  and  $r(k)$  according to their new rank in the tree. This may incur additional overhead. In the case of DTS-SS, when a node changes its parent, the expected send time and expected reception times are synchronized through one phase update when the node sends its first data report to the new parent. An advantage of DTS-SS is that it does not require any special mechanism for dealing with topology changes.

**Selecting timeout values.** In the case when packets are lost, either due to transient or permanent failures, a node may receive the data reports from a subset of its children. To avoid waiting indefinitely for the children's data reports, a parent times out and sends the aggregated data reports based on the ones it has received. For NTS-SS the timeout interval is set based on the node's rank:  $t_{TO}(d) = (d+1) * \frac{D}{M}$ . For STS-SS the timeout is set relative to the expected send time  $s(k)+l-t_{TO}$ , where  $l$  is the

local deadline and  $t_{TO}$  is a constant. For DTS-SS, since the time it takes a node to collect data from its children usually depends on the one-hop delay, the timeout is set to  $max_c(s(k,c)) + t_{TO}$ , where  $t_{TO}$  is a tunable parameter. The above timeout values are selected to balance the query latency and the effectiveness of in-network aggregation. A detailed discussion is omitted due to space limitations.

We observe that the traffic shaper affects the robustness of the protocols with respect to network dynamics. Since NTS-SS computes the expected reception and send times based on the properties of the query,  $(\phi, P)$ , which are independent of both the behavior of the neighbors and the tree topology, it is the most robust, i.e., it does not require any state update (except for the time out value discussed above) to deal with packet loss or topology changes. Since STS-SS computes the expected reception and send times based on a property of the tree topology, namely the rank of a node, it is not affected by transient loss of packets. However, in the case of a change in tree topology, reception and send times may need to be updated according to the new ranks. A benefit of DTS-SS is that it does not require any special mechanism for handling topology changes and requires only a new phase update to resynchronize the sleep schedules when data reports are transiently dropped.

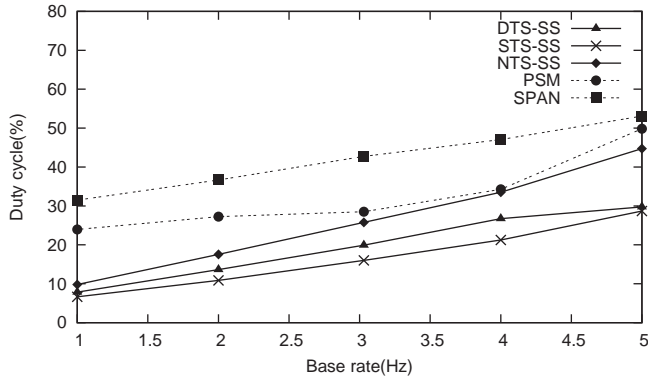
## 5 Experiments

Through *ns-2* simulations we evaluate the ESSAT protocols along three dimensions: energy efficiency, query performance, and the impact of the radio's break-even-time on energy efficiency.

In our simulations 80 nodes are randomly distributed in an area of 500 x 500  $m^2$ . The communication range is set to 125m. IEEE 802.11b is used as the MAC protocol. The network bandwidth is 1Mbps. Each data report is encapsulated in a single packet of 52 bytes.

We simulate three types of queries with different rates. The ratio of the rates of the three query classes  $Q_1 : Q_2 : Q_3$  is 6 : 3 : 2.  $Q_1$ 's rate is referred to as the *base rate*. When the base rate is varied, the rates of  $Q_2$  and  $Q_3$  also change proportionally. We vary the workload in two ways. First, there is a single query per class and the base rate is varied from 1Hz to 5Hz. Second, the base rate is fixed at 0.2Hz while the number of queries per class is increased. Each query starts at a random time chosen between 0 – 10s. All experiments last for 200s.

The root of the routing tree is the node closest to the center of the area. The root initiates the construction of the routing tree by flooding a setup request. Each node may receive setup requests from multiple nodes and selects the node with the lowest level as its parent. The routing tree is setup before the start of the experiments



**Figure 3.** Average duty cycle for three queries classes when varying base rate.

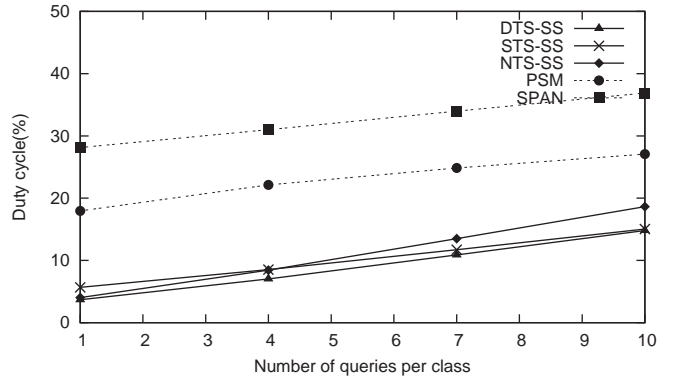
and spans all nodes located within 300m from the root. Each node in the routing tree performs in-network aggregation. We assume that each aggregated data report fits in a single data packet. STS-SS’s deadline is equal to its period.

For performance comparison we run several baselines: SYNC, SPAN and PSM. The SYNC protocol uses a fixed duty cycle, an approach adopted by synchronous wake up protocols [16]. All nodes share a synchronized periodic schedule. Each period includes fixed active and sleep windows. We configure SYNC to run at a duty cycle of 20% and a period of 0.2s. We chose a duty cycle of 20% to approximate the duty cycles of the ESSAT protocols in the case of medium workload. The period is set to be 0.2 seconds to coincide with the highest data rate (5Hz) of our experiments. We also chose PSM with the extensions proposed in [3] because it adapts to observed traffic through traffic advertisements. The beacon period, the ATIM window, and Advertisement window are set to 0.2s, 0.025s and 0.1s respectively. SPAN [3] is a power management protocol that uses a communication backbone. To reduce query latencies, the routing trees are modified such that all leaf nodes are sleeping nodes while non-leaf nodes are active nodes selected by SPAN. In the original SPAN protocol non-active nodes run PSM. In our experiments, the leaf nodes run NTS instead of PSM since it has better energy performance and lower query latency than PSM.

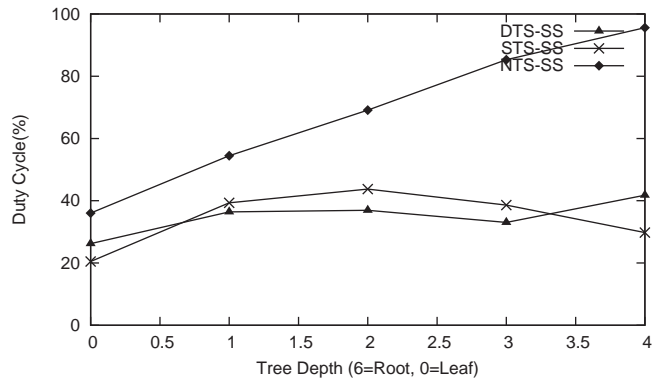
Unless mentioned otherwise, each data point is the average over five runs. The start time of each query and the node locations are varied in each run.

## 5.1 Energy Efficiency

We use average node duty cycle as a metric to evaluate the energy efficiency of the considered protocols. Figure 3 shows the impact of increasing the base rate of the queries on the average duty cycle of all nodes. For this graph, the 90% confidence intervals of all protocols are within  $\pm 2.3\%$ . SYNC is not shown in Figures 3 and 4 because every node is configured to a fixed duty cycle of 20%



**Figure 4.** Average duty cycle for three query classes when varying number of queries per class.

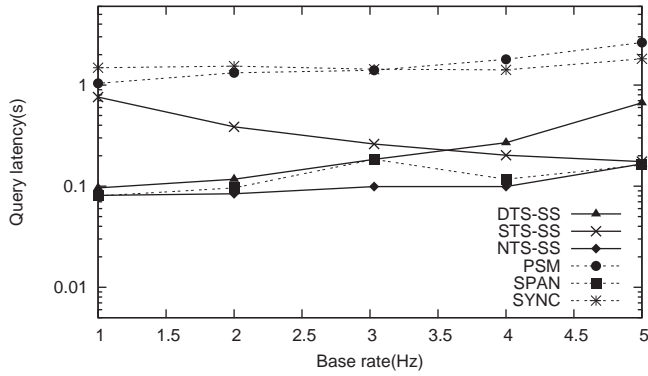


**Figure 5.** Distribution of duty cycles at different ranks.

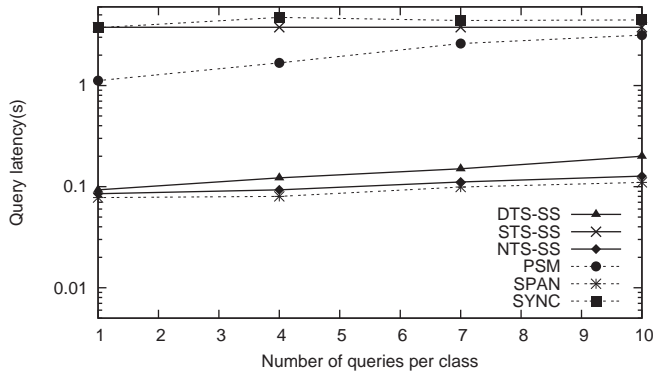
for all experimental settings. SPAN has the highest duty cycle in the network due to the high energy cost of maintaining a communication backbone. Since PSM does not maintain a communication backbone, it conserves more energy than SPAN. However, PSM transmits only overhead packets during the advertisement window incurring significant energy penalty. As a result, all ESSAT protocols have lower duty cycles than PSM. NTS-SS performs the worst among the ESSAT protocols. STS-SS has the best energy efficiency among the proposed protocols because its local deadline is longer than  $T_{agg}$  (see Equation 3). DTS-SS has a slightly higher duty cycle than STS-SS but remains consistently lower than NTS-SS. As the rate increases both STS-SS and DTS-SS increase their duty cycle to preserve the query performance (as shown in Section 5.2). These results are consistent with our analysis.

Figure 4 shows the average duty cycle when the base rate is 0.2Hz and the number of queries per class is increased. For this graph, the 90% confidence intervals of all protocols are within  $\pm 1.2\%$ . All ESSAT protocols again outperform the baselines. We note that DTS performs better in this experiment. Both Figures 3 and 4 show that DTS can effectively adapt to the workload without tuning.

As described in Section 4.1, a limitation of NTS-SS is



**Figure 6.** Query latency for three query classes when varying base rate.

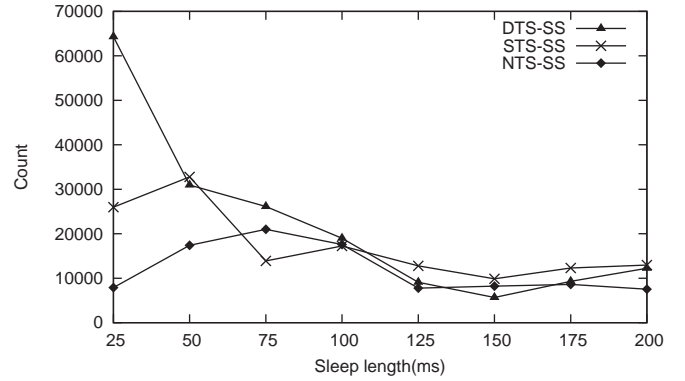


**Figure 7.** Query latency for three query classes when varying the number of queries per class.

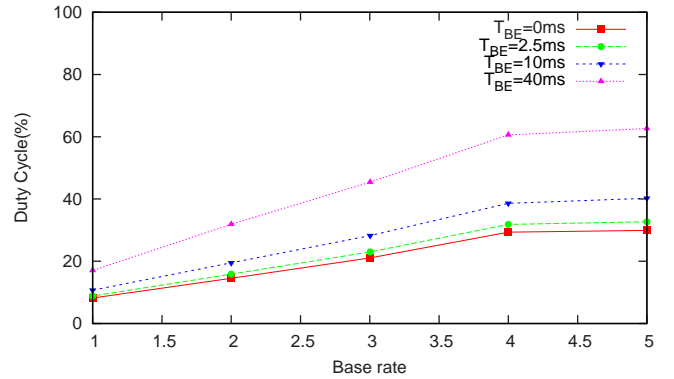
that it consumes energy unevenly. To validate our analysis Figure 5 plots the average duty cycles of the nodes with the same rank. The plot shows the experimental results from a typical run when each class has one query and the base rate is  $5Hz$ . As the node rank increases, there is a linear increase in the duty cycle. This is consistent with our analysis in Section 4.2.1. In contrast, the duty cycle of STS-SS and DTS-SS are independent of the rank of the node, making them more scalable. This result shows that STS-SS and DTS-SS distribute the energy consumption more evenly and can scale better to large routing trees than NTS-SS.

## 5.2 Query Performance

Figure 6 shows the average query latency as the base rate is increased. The 90% confidence intervals of ESSAT protocols and SPAN are within  $\pm 0.16s$  while for SMAC and PSM the 90% confidence intervals are within  $\pm 0.75s$ . NTS-SS and SPAN have the lowest query latencies. Since NTS-SS propagates the data reports greedily and wakes nodes up in time, it does not introduce any delay penalties. SPAN achieves small query latency by maintaining an active communication backbone. However, as shown in Section 5.1, the query latency of SPAN



**Figure 8.** Histogram of sleep intervals. Each point in the graph represents the number of sleep intervals whose length falls in the range  $[x - 25, x]ms$ .



**Figure 9.** Impact of distribution of sleep interval on STS-SS. Three queries are run at a base rate of  $5Hz$ .

comes at a high energy cost. All ESSAT protocols have significantly lower query latencies than SYNC and PSM. (Note the logarithmic scale of the figures). In SYNC and PSM the query latency is affected by the temporal relationship between the communication workload and their periodic sleep schedule. It is common for a data report to be buffered for considerable amount time resulting in higher query latencies. This problem is also reflected by the higher confidence of SMAC and PSM compared to that of the other considered protocols. As the local deadline of STS-SS is configured to equal the period of the query, its query latency decreases as the rate increases. In contrast, DTS-SS's query latency increases with rates. Despite this increase, the query latencies of DTS-SS are 36-98% lower than PSM and SYNC.

Figure 7 shows the query latency when the base rate is kept constant at  $0.2Hz$  and the number of queries per class is increased. The confidence intervals are similar to the previously discussed experiment. In contrast with the previous setup, the latency of the STS-SS is constant since the rate does not change. DTS-SS exhibits lower query latency than STS-SS.

### 5.3 Impact of Break-Even-Time of Radio.

As discussed in Section 4.1, fine grained power management needs to consider the costs of transitioning between power states. To quantify the impact of the break-even-time  $T_{BE}$  on energy efficiency of ESSAT protocols, Figure 8 plots the histogram of the sleep intervals when  $T_{BE} = 0$ . In light of Figure 8 we make two observations. First, based on the distribution, it is clear that the workload witnessed by nodes is aperiodic. Second, the impact of break-even-times must be taken into account. Otherwise a node suffers high penalties in query latency. For example, the percent of sleep intervals shorter than a break-even-time of  $2.5ms$  (typical wake up delay for MICA2's radio and WLAN) for NTS-SS, STS-SS and DTS-SS are 0.40%, 0.85%, and 6.33% respectively. This implies, that for DTS-SS, 6.33% of the times the node is turned off additional delay penalties would have occurred if the break-even-times had not been taken into consideration. A key advantage of SS is that it avoids such delay penalties as discussed in 4.1. Since DTS-SS is the most sensitive to break-even-times we plot duty cycles of DTS-SS when different  $T_{BE}$  values are used as parameters for SS. We carefully chose the values of  $T_{BE}$ . The 2.5ms and 10ms are the average and worst case break-even-times reported for MICA2's radio [6]. The  $T_{BE}$  of 40ms is reported in ZebraNet [7]. For  $T_{BE}$  values smaller than 10ms which are common to MICA2 motes, the duty cycle is increased by at most 10%. However, for  $T_{BE} = 40ms$ , an increase of as high as 30% is observed. This results confirm the importance of reducing the wake up time of radios in WSNs [6].

## 6 Conclusions

This paper presents ESSAT, a novel power management scheme that aggressively exploits the timing semantics of WSN applications. An ESSAT protocol is comprised of the Safe Sleep (SS) scheduler and traffic shaper (STS or DTS). A key feature of our ESSAT protocols is that they can conserve significant energy while introducing minimum delay penalties. For example, our simulations showed that DTS-SS achieved a average node duty cycles that are 38-87% lower than SPAN, and query latencies that are 36-98% lower than PSM and SYNC. Moreover, they can adapt to varying workload and network topologies at minimum overhead. As a result, ESSAT is especially suitable for WSNs that operate under both power and timing constraints in dynamic environments. In the future, we plan to integrate our protocols with existing query services on a testbed of MICA2 motes.

## References

- [1] Wireless lan medium access control (mac) and physical layer (phy) specifications. *IEEE Standard 802.11*.
- [2] L. Benini, A. Bogliolo, and G. D. Micheli. A survey of design techniques for system-level dynamic power management. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 8(3), 2000.
- [3] 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.
- [4] B. Hohlt, L. Doherty, and E. Brewer. Flexible power scheduling for sensor networks. In *IPSN 2004*.
- [5] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *MobiCom*, 2000.
- [6] C. S. D. C. Joseph Polastre, Robert Szewczyk. The mote revolution: Low power wireless sensor network devices. In *Hot Chips 16*, 2004.
- [7] T. Liu, C. M. Sadler, P. Zhang, and M. Martonosi. Implementing software on resource-constrained mobile sensors: experiences with impala and zebranet. In *MobiSys*, 2004.
- [8] S. Madden, M. Franklin, J. Hellerstein, and W. Hong. Tag: a tiny aggregation service for ad-hoc sensor networks. In *OSDI*, 2002.
- [9] V. Rajendran, K. Obraczka, and J. J. Garcia-Luna-Aceves. Energy-efficient collision-free medium access control for wireless sensor networks. In *Sensys*, 2003.
- [10] J. Sun. *Fixed-Priority End-To-End Scheduling in Distributed Real-Time Systems*. PhD thesis, UIUC, 1997.
- [11] Y.-C. Tseng, C.-S. Hsu, and T.-Y. Hsieh. Power-saving protocols for ieee 802.11-based multi-hop ad hoc networks. In *INFOCOM*, 2002.
- [12] T. van Dam and K. Langendoen. An adaptive energy-efficient mac protocol for wireless sensor networks. In *Sensys 2003*, 2003.
- [13] X. Wang, G. Xing, Y. Zhang, C. Lu, R. Pless, and C. D. Gill. Integrated coverage and connectivity configuration in wireless sensor networks. In *Sensys*, 2003.
- [14] A. Woo and D. E. Culler. A transmission control scheme for media access in sensor networks. In *MobiCom*, 2001.
- [15] Y. Xu, J. Heidemann, and D. Estrin. Geography-informed energy conservation for ad hoc routing. In *MobiCom*, 2001.
- [16] W. Ye, J. Heidemann, and D. Estrin. An energy-efficient mac protocol for wireless sensor networks. In *INFOCOM*.
- [17] J. Zhao and R. Govindan. Understanding packet delivery performance in dense wireless sensor networks. In *Sensys*, 2003.
- [18] R. Zheng, J. C. Hou, and L. Sha. Asynchronous wakeup for ad hoc networks. In *MobiHoc*.
- [19] R. Zheng, R. and Kravets. On-demand power management for ad hoc networks. In *INFOCOM*, 2003.