

# Adaptive Ad-hoc Self-Organizing Scheduling for Quasi-Periodic Sensor Network Lifetime<sup>\*</sup>

Sharat C. Visweswara<sup>a</sup>, Rudra Dutta<sup>a,\*</sup>, Mihail L. Sichitiu<sup>b</sup>

<sup>a</sup>*Computer Science department and*

<sup>b</sup>*Electrical and Computer Engg. department, North Carolina State University*

---

## Abstract

Wireless sensor networks are poised to revolutionize our abilities in sensing and controlling our environment. Power conservation is a primary research concern for these networks. Often, the single most important savings can be obtained by switching off the wireless receiver when not needed. In this paper, we describe an algorithm which allows the nodes to learn the behavior of each other by only observing the transmission behaviors, and from this derive the schedule without external help. Our approach is robust to statistical variations in the nodal transmission periods. We draw important conclusions on the effect of quasi-periodicity on the scalability of the solution. We provide results of numerical simulations that show the effectiveness of our approach.

### *Key words:*

Sensor networks, lifetime, optimization, scheduling, lifetime extension, power conservation, power aware sensor networks

---

## 1 Introduction

In recent years, advances in wireless communication and sensing technology has made it possible for a very large number of individually cheap and small units, combining sensors and wireless network nodes, to be used in *wireless*

---

<sup>\*</sup> This research was supported in part by National Science Foundation grant # EEC-0332271. Some preliminary results from this research were presented at *SECON* 2004.

<sup>\*</sup> Corresponding author. Room 446, Monteith Research Center, North Carolina State University, Campus Box 7534, Raleigh, NC, 27695-7534

*Email address:* [dutta@csc.ncsu.edu](mailto:dutta@csc.ncsu.edu) (Rudra Dutta).

*sensor networks*. Sensor networks are a class of *ad hoc wireless networks*, and as such are expected to be deployed with a minimum of *a priori* assignment of network roles (such as routing), forming a so-called *instant infrastructure* after deployment. There has been significant recent interest in the literature on all aspects of ad hoc networks in general and sensor networks in particular, good recent surveys of the literature can be found in [1, 2].

Sensor networks are typically multihop like general ad hoc wireless networks, and some nodes forward traffic from other nodes as well as originating their own data traffic. A large number of sensor networks in practice share the following salient characteristics: (i) a very large number of nodes, (ii) an *egress* traffic pattern, (iii) traffic with low volume and high regularity, (iv) low or no mobility, and (v) a critical sensitivity to battery power efficiency.

The sensing application which motivated our study was the structural health monitoring of bridges, buildings, and other large civil structures, *i.e.* the continuous monitoring of strain and related phenomena in steel or concrete members forming these structures. All sensors transmit their readings as data packets to a single *monitoring station*, usually located in the basement or abutment of the structure. It conforms to all the characteristics mentioned above. Practical experience shows that many sensors are damaged or disabled during or shortly after deployment in such networks. Accordingly, we have considered it appropriate to continue only with the assumption of reasonably uniform density of sensors over the sensing field. As such, our results are widely applicable to many typical sensing applications employing *continuous sensing*, *i.e.* in which each sensor periodically produces a constant small amount of data.

### 1.1 Prior Work

Power conservation is obviously of great importance in such an application, since the network is expected to be in operation for at least the expected lifetime of the building or other structure, which is of the order of decades. The efficient use of power is generally a very important research area in sensor networks, and there has been significant recent attention to this problem in the literature. Many aspects of the problem have been extensively studied [3–26]. Some approaches [3–5] aim to manipulate the power level of the transmitter, while still maintaining network connectivity; thus these approaches operate at the physical layer. Others focus on medium access control (MAC) layer techniques [6–16] to conserve battery power by turning the receiver off whenever it is not needed (e.g., when a node detects an ongoing conversation). Yet others focus on routing [17, 18], proposing algorithms to choose routes in such a way as to maximize the lifetime of the network. For applications other than sensing, approaches at the application layer may be highly appropriate

as well. While simply combining such strategies is likely to produce benefits that exceed those of the individual strategies, it seems likely that the power problem is best addressed by integrating the consideration of more than one layer in the design [24–26].

There are many components to the power expended by sensor network nodes. In attempting to lengthen network lifetime, the most beneficial component to address must be chosen depending on the application. Very often the power consumption component that turns out to be of overarching importance is the *idle listening power*; this is true of the class of applications we consider as well. Our basic approach, therefore, is to allow the nodes to cooperatively discover a *schedule* of transmissions and receptions, and thus allow them to turn off their wireless transceivers (and possibly microprocessors) off for significant parts of their duty cycle.

## 1.2 Our Contribution

The idea of scheduling nodes for switch-off or power-down to extend network lifetime has been explored in the literature before. In particular, we have presented previous work on this issue in [26]. In that paper also, an approach to obtain a schedule of transmission and reception is described, allowing the nodes to turn off their transceivers to save power. Similarly, many of the aforementioned literature on power-conservation propose switching off the radio for certain periods by control at the MAC layer, effectively creating a schedule of switch-on and switch-off for all nodes in the network (but not necessarily obtaining the schedule from a global perspective). Such actions are even part of deployed protocols such as IEEE 802.11.

However, our contribution in this paper is completely new. Briefly, in [26] we assumed (similar to many other works in literature) that nodes are either perfectly synchronized, or synchronization can obtain a hard bound on the disagreement between clocks at different nodes. Thus the consequence of disagreement between clocks can be addressed by the introduction of a “guard period”. While this is a useful scenario, there is also the possibility that no such hard guarantee on clock disagreement or drift can be obtained, as we explain in detail in the next section. In such a case, existing methods would require large guard periods (reducing the lifetime gain from sleeping) or lose large amounts of traffic. In Section 3, we show why this approach will fail in the face of a slight but continuing drift between clocks.

Accordingly, we must explicitly represent the uncertainty in the periodicity of the nodal transmissions. We refer to this as *quasi-periodic* traffic. This difference requires our actual approach to be quite different. In [26], an initial

configuration period of exchanging control signals between nodes is used to form the schedule. In contrast, we follow an adaptive approach in which each node learns the information it needs by simply observing the behavior of its neighbors. This implies that a node can track the changing behavior of a neighbor, and do this without requiring other nodes in the vicinity to abandon their schedules. Our contributions also include an insight into the effect of quasi-periodicity on the scalability of this problem, and the need for traffic shaping. Further, we show that the concept of network lifetime has to be broadened under these circumstances. In our numerical results, Figures 11(a)-11(b) and following show that our approach, responsive to quasi-periodicity, does better than an approach which is similar except that it does not take quasi-periodicity into account.

The rest of the paper is organized as follows. In the next section, we define the problem more fully, and state our assumptions. Section 3 presents our insight that quasi-periodicity requires traffic shaping by the nodes. Section 4 describes our scheduling algorithm in detail and articulates various options and variations in our basic approach. We have actually coded this algorithm and run extensive numerical simulations; we present the results of those simulations in Section 5. Finally, Section 6 concludes the paper.

## 2 Assumptions and Problem Definition

The behavior of a node in putting itself to sleep can be characterized in the broadest terms as an alternation between two states:

- (1) **Awake:** Wait for an incoming transmission. Once the transmission is received, compute the next sleep period, using some scheduling algorithm. Set the off-chip sleep timer for this period, and go to Sleep State by switching off the transceiver and the processor.
- (2) **Sleep:** When sleep timer expires and wakes up the processor, power up the transceiver and go to Awake State.

In this section, we discuss the issues involved in designing a scheduling algorithm that is appropriate for use by the above algorithm. We make several assumptions about the problem, all of which are reasonable and stem from practical considerations. In terms of modeling the transmission and receptions, we focus entirely on the network layer. We assume some working MAC layer exists underneath, probably based on RTS/CTS mechanisms. The delays introduced by such a protocol are absorbed in the uncertainty we use for the inter-arrival times, as mentioned below.

Currently and in the foreseeable immediate future, sensor networks that are the focus of this paper (unattended, power-limited, desired long lifetime) are likely to employ simple nodes with single radios operating on one channel. Our current work focuses on such single channel networks; similar scheduling approaches for multi-channel multi-interface nodes is part of our ongoing work, but out of scope of this paper.

## 2.1 Stochastic Inter-arrival

Since we are considering continuous monitoring sensors, we assume that each node is meant to produce a regular stream of data. However, realistically the interval between successive transmissions from a node will not be precisely constant, variations will occur due to several factors. The timer that a sensor uses to determine sensing epochs is bound to realistically have some inaccuracy, as well as some drift. A node that takes a sensor reading and attempts to transmit will be delayed by RTS/CTS mechanisms at the MAC layer by a variable amount of time. The sensor itself may take different amounts of time to complete taking the reading after it is triggered by the timer. We must also acknowledge that there may be other factors in the environment that introduce random variations in the period. Note that clock synchronization is not the answer here, because randomness is also injected by factors other than the clock; and anyway a synchronization algorithm could only operate during the times that a node was awake, at which time the damage is already done. A clock synchronization algorithm may help against long-term divergence of the period, but not against the variations in successive inter-arrival times. Now the inter-arrival time between successive transmissions received from a neighbor must be modeled as a stochastic quantity, varying according to some Probability Density Function (PDF), rather than constant (*i.e.* an impulse PDF).

### 2.1.1 Nature of PDF

Rather than make strong assumptions about the nature of the PDF, we attempt to define it minimally and keep our approach general and robust. Our sole assumptions are that: the PDF is peaked, with a single mode. The assumption of unimodality is quite usual and reasonable. The assumption of peakedness is not essential to our approach, but the benefit that can be obtained by our approach (or indeed, any approach) will diminish as the PDF becomes more platykurtic and the uncertainty in the period increases. For this reason, we focus our attention on PDFs with high kurtosis. However, we do *not* make the less realistic assumption that the PDF is bounded and these bounds are known *a priori*. In other words, the PDF, though light-tailed, can

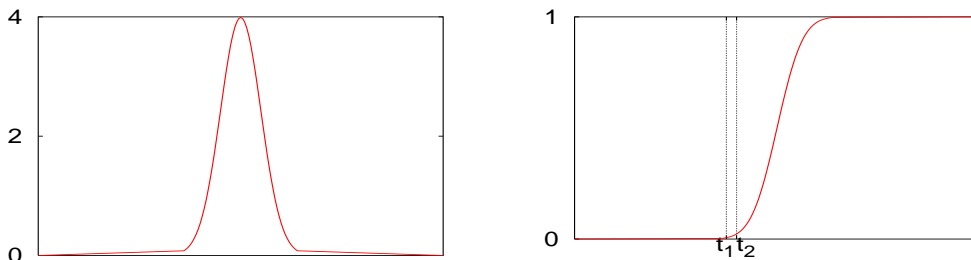


Fig. 1. A light-but-long-tailed peaked PDF, corresponding CDF

have a long tail that extends almost to zero. Such a possible PDF is shown in Figure 1, with the corresponding Cumulative Density Function (CDF).

### 2.1.2 Independence

Since the processor may itself be put to sleep using this scheduling approach, it is appropriate to think of a given node as having no memory of its own past transmission instants. In other words, successive inter-arrival times from that node are independent random variables, each with the same PDF. The alternative is to assume that while the node is woken up at intervals that vary, once it is awake it can precisely determine that variation. This requires the ability to refer to a perfect clock; assuming that the node has no such clock on-board, it can only appeal to a perfect “superclock” that is universally available to all nodes, *e.g.* one supplied by a GPS system. Studying the system under such conditions is out of the scope of this paper. Realistically, such is not available in many sensing applications due to the presence of foliage, indoor location or enclosure by metal structures.

### 2.1.3 Non-stationary PDFs

Later in our paper, we extend our approach to cases where the PDF of the inter-arrival times is itself not stationary but goes through periods when the mean slowly drifts. This type of situation may arise due to clock drift before it is corrected, and it would be desirable for the scheduling algorithm to be robust to it. However, it is important to note that the mean transmission period of a node may also change due to the requirements of the sensing application - such changes are likely to be quite abrupt. For example, the base station may send a control message to a node requesting a higher frequency of sensor readings, or a sensor node may have been pre-programmed to switch to double frequency whenever the sensed quantity exceeds a particular value (“temperature > 100 C”, say). In all such cases, the node would be aware of this change, and it would be a simple protocol-related issue to warn the downstream nodes of the change. We do not address such changes.

## 2.2 Metric for Benefit: “Utility”

An important issue in this context is the measure of the benefit offered by the scheduling. If the service provided by the network to the sensing application (transport of sensor readings) remained exactly the same, then the increase in lifetime of the network would be a good measure of the benefit of scheduling, and could be inferred easily from the fraction of time the network nodes were able to sleep. However, for the light-but-long-tailed type of PDF we mentioned above, over time some transmissions will be lost for almost any non-zero amount of sleep. In other words, the use of “guard periods” to prevent loss will result in the nodes hardly sleeping at all and therefore no benefit to the network. However, if we allow the nodes to sleep longer, we must resign ourselves to some loss. In that case, simple lifetime no longer suffices as a good measure of the benefit; we could attain infinite lifetime by putting the nodes to sleep forever. The loss is 100%; the network lives forever but is of no use to the sensing application. Instead of setting arbitrary levels of acceptable loss (which are properly considered to be specific to the sensing application), we focus on a measure of the benefit that is robust to such situations (as well as that of a nicely bounded PDF). We assume that the primary purpose of the sensing application is to transfer readings to the base station, and we define the **utility** of the network as the *total number of readings transferred from the sensors to the base station over the lifetime of the network*. Thus a small loss, if it can be leveraged into a large increase in lifetime, will result in an overall improvement in utility, but larger losses will reduce utility even if the lifetime is improved. In Section 4 we discuss this further. Note that utility continues to accrue while individual nodes of the network die but the network remains connected and some sensor readings continue to be received at the monitoring station. In the language of reliability studies, this reflects the *performability* of the network viewed as a system, rather than any specific measure of performance [27, 28].

With all the above in mind, we can identify our problem as that of designing an algorithm which enables each node to compute sleep periods by observing the transmission characteristics of its neighbors, adapting to changes in such characteristics, such that a high value of utility results for the network as a whole.

## 3 The Need for Traffic Shaping

We turn our attention now to the algorithm according to which nodes transmit. Every sensor node generates a data flow corresponding to its own sensor readings. In addition, a sensor node will in general be used as next-hop router

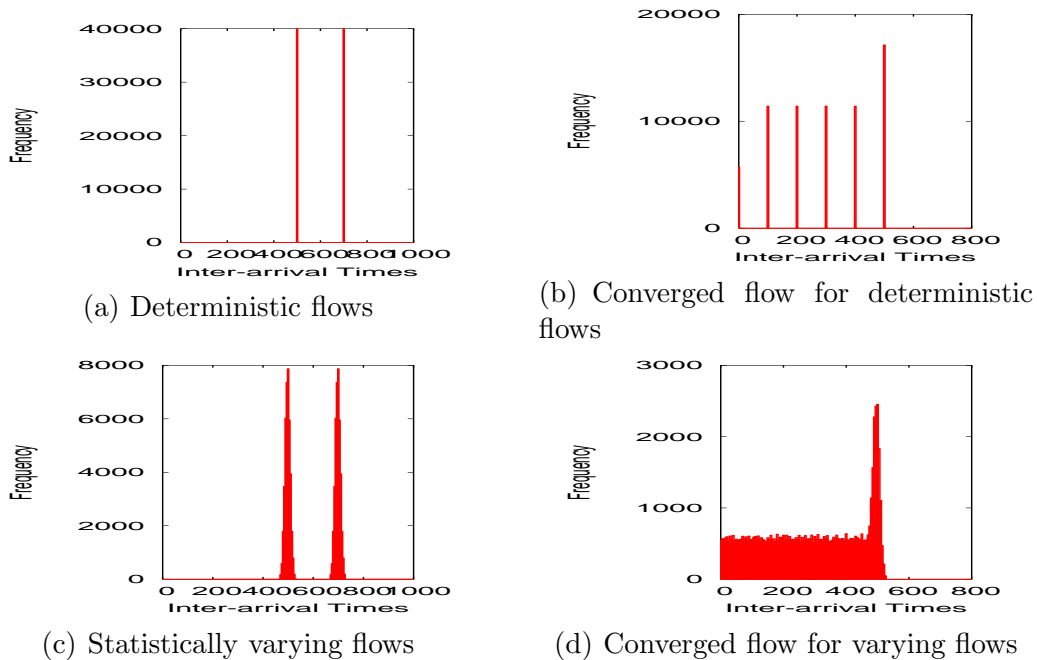


Fig. 2. Merging two flows. Two flows with impulse interarrival time distribution result in distribution comprising multiple impulses (above). Two flows with some uncertainty in interarrival time distribution result in complete loss of certainty over broad region in output distribution (below).

by other nodes, and these are flows that this node must forward. The natural approach would be to forward every transmission as soon as it is received; below we show that this straightforward approach unfortunately does not work in the presence of uncertainty.

In what follows we refer to the direction towards the monitoring station as *downstream* and the reverse direction as *upstream*. Thus a node forwards traffic from some of its upstream neighbors to the downstream neighbor that is its next-hop router. Consider a node  $A$  that forwards traffic from two upstream neighbors to the downstream node  $B$ . When the inter-arrival time seen by  $A$  from each upstream neighbor is strictly periodic, and  $A$  forwards each transmission as soon as it is received, the aggregate flow seen by  $B$  is also characterized by a combination of impulse PDFs. This situation is illustrated in the top part of Figure 2. Thus  $B$  can still discern periodicity in the behavior of  $A$  without knowing about  $A$ 's upstream neighbors, and may be able to use this information to form a sleep schedule. However, the situation changes dramatically when  $A$ 's upstream flows are only quasi-periodic, as shown in the bottom part of Figure 2. As we see, the PDF of the input flows is now slightly spread out – note that it is still strictly bounded, not infinite-tailed or even particularly long-tailed. However, the PDF of the flow seen by  $B$  has lost all structure except a spike at the end. The spike is quite tall, but contains comparatively little of the density of the PDF, the bulk of which is now in the nearly-uniform spread preceding the spike.

This result may seem surprising at first, but it is obvious on reflection. Remember that successive inter-arrival times are independent by our assumption. In that case, even a slight uncertainty will cause small deviations from the exact phase relationship of the two flows, *which can then accumulate*. This follows directly from our assumption about the independence of successive inter-transmission times for a node (equivalently, the unavailability of a perfect reference clock). As a result, very shortly, any initial phase relationship completely disappears, and over time no particular relationship can dominate. The figures were produced by actual numerical simulation of merging two flows, with about 100,000 arrivals from the more frequent of the two sources.

This affects our algorithm in two important ways. First, even if all the nodes generate flows with PDFs that conform to our peaked requirement, this will not survive even one hop. To enable the node  $B$  in the above example to form a sleeping schedule, it must have PDF information not just for  $A$  but individually for every node whose traffic is routed through it. Clearly, in a sensor network where the number of nodes is in the thousands or tens of thousands, this is impractical from several points of view. For one thing, node  $B$  must obtain and store PDF representations for a very large number of nodes, and use them in computing the schedule - this is likely to be beyond the capacity of the simple processor on the sensor node. More importantly, even with perfect information and perfect processing, the windows of opportunity during which the node might sleep will clearly become very fragmented when a large number of sources are originating flows. We are drawn inescapably to the conclusion that  $A$  cannot be allowed to forward transmissions as soon as they arrive; it must instead *shape* traffic so as to present a (quasi-)periodic flow to  $B$ . One obvious period for  $A$  to choose is that corresponding to its own sensing function, we discuss alternatives in the next section. The second conclusion is that each node must nevertheless have individual PDFs for at least each of its upstream neighbors. In the example above,  $A$  would be in the same position as we articulated for  $B$ , if it did not individually recognize the PDFs for its two upstream neighbors. However, this does not pose a large problem, because with uniform density of sensors, every node can be expected to have only a small number (no more than 3 or 4) of upstream neighbors.

There is a final point with respect to traffic shaping. We have been referring to transmission instants, implicitly assuming that the transmission delay (but not necessarily the propagation delay) of sensor data packets is negligible. With traffic shaping, a node must store incoming transmissions until the next transmission instant (as determined by the shaping) arrives - that is in-network aggregation is performed. This requires buffer space and will make the next transmission from this node larger, since several stored sensor readings must all be sent at once. However, we can continue with our earlier model, since typical sensor applications show that the sensed data may be as small as 2 bytes, while the overhead per transmission may be 6 bytes. Even with stored

data from each of the small number of upstream neighbors, the transmission time at typical wireless bitrates will still remain very small compared to the period, which may be half a second or one second.

## 4 Algorithm Description

To solve the problem as defined above, the algorithm's primary goals are to shape its traffic and learn the shape of the traffic from its upstream neighbors. Some other properties are desirable: the algorithm should stabilize quickly and be able to adapt to changes quickly. Some applications are time sensitive and may require that sensor data reach the monitoring station within a specified time to be useful. The algorithm should respect such delay bounds if they are specified.

Our algorithm is distributed, i.e. it runs on every node in the network and has no centralized component. The algorithm requires a parameter: the fraction of loss that is acceptable at each hop, which we shall refer to as the *hop-loss* parameter  $k$  in the following text. A delay bound may also be provided.

### 4.1 Core Algorithm

The algorithm revolves around the three basic functions of the sensor node: generating sensor data, receiving transmissions from upstream neighbors and transmitting to downstream neighbors. We proceed under the assumption that the environment is sensed and data is generated at regular intervals. To shape traffic, generation is decoupled from transmission, i.e., nodes do not transmit whenever data is generated. Instead data are added to a buffer, the entire contents of which will be sent in the next transmission and the buffer emptied.

#### 4.1.1 Representation of Traffic Shape

In our algorithm, by traffic shape, we mean the probability distribution of inter-arrival times. This is represented in a histogram in which the range of inter-arrival times is divided across several bins and each bin contains the count of observed inter-arrival times that fall in that subrange. Since we are always interested in the *current* shape, we use a histogram constructed only from recent observations. We use the term *cycle* to mean the number of recent observations that are used to construct the histogram. If the cycle is too short, the shape will not be accurate. If it is too long, the algorithm will be slow to detect and adapt to changes.

The most useful value to use for the cycle is likely to depend on the size of the network, the number of maximum hops from the outer nodes to the base station, and other factors. In the scope of this paper, we have not considered the problem of obtaining the best cycle value. For our numerical simulations, we experimented over a large range of values for the cycle, and chose the minimum value at which further increase in the cycle does not result in better performance of the algorithm. This turned out to be a few hundreds (which is likely to represent only a tiny fraction of the total number of transmissions any node makes in the lifetime of the network). Since better initial estimation of the histogram makes for better performance, we conjecture that this value is very near the best value for the simulation scenarios we considered.

#### *4.1.2 Three Phases of the Algorithm*

The learning and shaping of traffic proceeds in three phases. In the first phase, the node is passive. It determines its transmission period based on the end-to-end delay constraint, if provided, and its data generation period. It is receiving data from upstream neighbors, and is recording inter-arrival times in order to construct the shape of each upstream node's traffic. It is not yet using these shapes to shape its own traffic and it is not sleeping. This phase lasts less than a cycle and should be just long enough to obtain a good estimate of the mean inter-arrival time of its upstream neighbors.

In the second phase, the node starts actively shaping its traffic in response to the traffic of its upstream neighbors - it has sufficient data for this purpose. Algorithm 1 shows how the node determines its own transmission period from the shape of the traffic of its upstream nodes. This phase is meant to allow the node to adapt its shape and by the end of the phase, it must be stable. The node cannot sleep in this phase as its upstream neighbors will themselves not be stable until the end of this phase. The effect of Algorithm 1 will ripple across the network from the outer to the inner nodes. The time taken for the entire network to stabilize depends on the depth of the network. A cycle after a node has stabilized, its downstream neighbor will have a sufficiently accurate idea of its traffic and will be able to sleep. Hence this phase must last at least two cycles.

In the third phase, nodes may sleep. After every reception, the node determines which upstream neighbor will transmit next and until what time it can sleep, using Algorithm 2. This algorithm utilizes the allowable loss parameter  $k$  mentioned above to calculate a value  $t_k$ . Then,  $t_k$  and the time of the last arrival is used to determine when the next transmission is likely to arrive. The node will then stay awake until a transmission from that neighbor is received. In this phase, the node may also attempt to detect changes in the behavior of its upstream neighbors and adapt to these changes. We will describe in detail

three strategies that can be used to accomplish this later in the paper.

---

**Algorithm 1** Calculate Transmission Period (Phase 2)

---

```

{ $n.d_{max}$ : node  $n$ 's delay bound}
{ $n.d$ : node  $n$ 's depth}
{ $n.t_{gx}$ : node  $n$ 's data generation period}
{ $n.t_{tx}$ : node  $n$ 's mean transmission period}
{ $U$ : set of all upstream neighbors}
 $d_{max} \leftarrow \min\{u.d_{max}\} \forall u \in U$ 
if  $self.d_{max} < d_{max}$  then
     $d_{max} \leftarrow self.d_{max}$ 
end if
 $t_{tx} \leftarrow \min\{u.t_{tx}\} \forall u \in U$ 
 $self.t_{tx} \leftarrow \min\{d_{max}, t_{tx}\}$ 

```

---



---

**Algorithm 2** Calculate Sleep Time (Phase 3)

---

```

{ $S$ : shape, treated as c.d.f}
{ $last$ : last arrival time}
{ $k$ : allowable loss factor}
{ $ETA$ : estimated time of arrival of next tx}
 $min \leftarrow \infty$ 
for each  $un$  do
    find  $t_k \mid un.S(t_k) = k$ 
     $ETA = un.last + t_k$ 
    if  $ETA < min$  then
         $min \leftarrow ETA$ 
    end if
end for
if  $min < now$  then
     $sleep \leftarrow 0$ 
else
     $sleep \leftarrow min - now$ 
end if
return  $min$ 

```

---

#### 4.1.3 Delay Bounds and Transmission Periods

We mentioned above that some applications are time sensitive and may have end-to-end delay constraints. We also mentioned that in the very first phase of the algorithm, the node computes its transmission period based on this constraint. If data from the node must reach the monitoring station in less than a given time period, then this period is divided equally across every hop on the path to the station to obtain a hop-by-hop delay bound. The node then uses the hop-by-hop bound as its transmission period, which ensures that the actual

*end-to-end delay* is less than or equal to the one specified. This may result in situations where a node must transmit in order to maintain periodicity, but has nothing to send. In such a case, the node will send a *dummy* transmission, which is empty of data. The transmission period thus calculated is not allowed to exceed the period of data generation. This minimizes the amount of buffer space needed on the sensor node and also ensures freshness of data.

An example may help clarify this. Consider a node  $A$  that has a sensing period of 1 second, but forwards data from a node  $B$  for which the per-hop delay bound is only 0.5 second. If  $A$  only makes a transmission once a second, one out of every two transmissions from  $B$  will miss its delay bound. Thus  $A$  must update its period to 0.5 second. On the other hand, if  $B$ 's per-hop delay bound is 5 seconds,  $A$  can transmit once every 5 seconds from the point of view of delay bounds, but this causes  $A$  to hold its own sensor data for upto 4 seconds. Since this is likely to be a concern from considerations of the freshness of data,  $A$ 's transmission period is never allowed to exceed its own sensing period of 1 second.

Note that this will result in the actual end-to-end delay reaching the allowed delay bound only for some nodes. For others, the end-to-end delay will be strictly less than the bound, and some of the allowed delay will remain unutilized. This is the only possible approach if the delay bounds are hard (*i.e.* they must be met absolutely); since upstream nodes have no control over the exact phase of the downstream nodes, any attempt at more closely utilizing the delay bound could result in the violation of the delay bound for the data from the worst-case node.

Specifying no delay bound is equivalent to specifying an infinite delay bound, in which case the transmission period is at most the data generation period because of the constraint described above. Note that this implies that *dummy transmissions are never required* unless delay bounds are being used. In Section 5 we investigate the effect of delay bounds on dummy transmissions. These results also show that the tradeoff between using dummy transmissions and energy savings is favorable in the case of bounded delays.

It is possible to attempt to distribute the delay bound unevenly across hops; in particular, nodes closer to the base station are likely to define the lifetime of the network and allowing them a higher proportion of the delay bounds is likely to provide better results. However, such an algorithm will be significantly more complicated, since a node must take into account its own depth when computing delay bounds, and this depth may change over time (as nodes die and routing is updated). Such a study is likely to be useful; however, it is out of scope of the current work.

## 4.2 Adaptation Strategies

We now describe the third phase of the algorithm in more detail. Activity in this phase depends on how the node attempts to detect and adapt to changes in the behavior of its upstream nodes. Any such strategy is complicated by the fact that observed inter-arrival times may not be valid. Since the node is sleeping, it is missing some transmissions. When one transmission is missed and the next one received, the observed inter-arrival time is very large. If such values are used in the histogram, the resulting shape will be distorted. There will be no density to the left of  $t_k$  and the larger inter-arrival times observed after missed transmissions will add density elsewhere. Therefore the node will actually obtain increasingly incorrect estimations of the PDF if it continues to record the interarrival times. Below we discuss three possible approaches to this situation that we study, the first being the trivial approach of ignoring any change (this serves as the base case with which to compare the other two, more active, approaches).

### 4.2.1 Non-adaptive - Base Case

In this approach, the node neither observes nor adapts. It knows the current behavior of its upstream nodes and assumes that they continue to behave the same way indefinitely. At the very beginning of phase three, the result of Algorithm 1 is stored and used; the algorithm is never run again. The shapes are also frozen, the histograms are never updated. This strategy avoids distorting the histogram after the node starts sleeping, but cannot adapt to any change in the behavior of upstream neighbors. If that behavior does not change then this strategy is very effective. It also provides us a basis for evaluating the advantage of the other two strategies that do adapt.

### 4.2.2 Using Sequence Numbers

If every upstream neighbor attaches a sequence number to its transmissions, then the node will know when it has missed one. The observed inter-arrival time can then be divided equally to obtain the successive inter-arrival times. They can be adjusted, if necessary, to ensure that the missed packet(s) arrived before the node woke up from its sleep. These values can be used in the histogram, providing a reliable histogram in spite of missed transmissions. This means that the node can continuously adapt to changes in upstream neighbor behavior.

### 4.2.3 Without Sequence Numbers - “Mode Watching”

It may not always be possible to use sequence numbers. Very simple sensor nodes may not be imbued with identity, making it impossible to associate a sequence with a node. Sequence numbers may be too large an overhead for small sensor data packets. We provide an algorithm to detect and adapt to changes even without sequence numbers.

As in the non-adaptive case above, the node stores and uses the result of Algorithm 1. Unlike in the non-adaptive case, the node is observing its upstream neighbors and updating its histograms in the third phase. Without sequence numbers, we are not accounting for missed packets, so we know that the resulting shape is distorted and unreliable. Hence,  $t_k$  of each neighbor as calculated in Algorithm 2 is stored and used.

We assert that the *mode* of the histogram is a property that will not be changed by the distortion caused by sleeping. To see this, we have to consider the transformation the PDF undergoes due to the missed transmission by a downstream node. We have proceeded with an analytical treatment of this question, but the expressions for the transformed PDF in terms of the input PDF involves a series of integrals which is not amenable to a closed-form representation, and does not provide any insight into the transformation. Instead we offer the following qualitative discussion, which captures the essential nature of the transformation. Consider the PDF of inter-arrival times perceived by a node which is sleeping as dictated by Algorithm 2. Obviously, no probability density can be perceived at any point to the left of  $t_k$  by such a node. This missing density is redistributed to the far right of  $t_k$  by combination with the interarrival time of the first packet that the node receives after waking up. Since the dominant effect will be that of exactly one packet being lost during the sleep (with sharply decreasing contributions from the cases of exactly 2 packets being lost, exactly 3 packets being lost, etc.), the result, to a good approximation, will be to superpose a small replica of the original PDF on the far right of  $t_k$ . More importantly, the nature of the PDF in the near vicinity of  $t_k$  will suffer no highly localized effect, because the change of density will be spread out. This situation is illustrated in Figure 3, which was obtained by actual numerical simulation (with an exaggerated variance for the input PDF). Thus the mode of the resultant PDF will not be shifted by the perceived change of PDF due to missing packets during sleep, and will remain in the same place as for the input PDF. Therefore the mode will shift only if the behavior of the upstream node changes. The above depends on the very reasonable assumption that  $k$  is much smaller than the position of the mode, which is 50% for a symmetrical PDF.

Thus the node may monitor the mode of the histogram and if it changes by a significant amount, the node concludes that the neighbor being observed

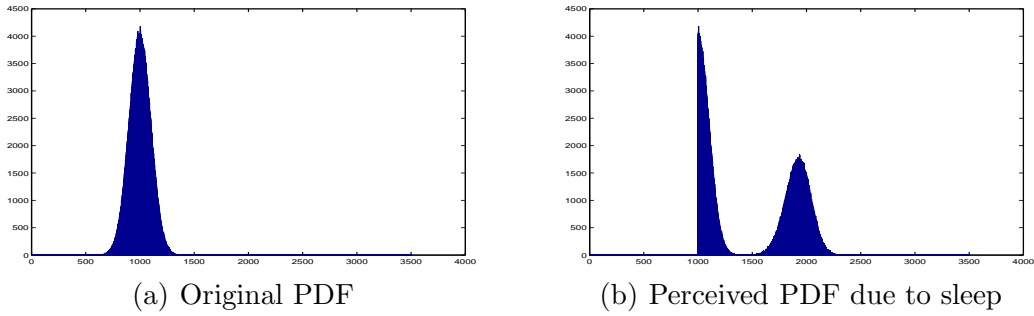


Fig. 3. Distortion of the PDF as perceived by a node executing Algorithm 2 does not cause mode shift

has changed its behavior. The node reacts by switching to phase one of the algorithm. It will try to learn the new behavior from scratch, before attempting to sleep again. Because the node “watches” the mode of the histogram for each upstream neighbor and reacts when the mode changes value, we refer to this adaptation approach as “mode watching”.

### 4.3 Expected Behavior

Performance of the algorithm and the adaptation strategies can be measured with four metrics: the fraction of time the node is asleep (sleep), the fraction of the node’s data that is lost on the way to the monitoring station (loss), the average delay between data generation at the node and the data reaching the monitoring station (delay) and the ratio of the number of transmissions to the number of sensor readings forwarded (overhead).

Sleep and loss together determine utility of the network, and the rest of the paper focusses primarily on these. Delay and overhead are factors that are relevant mainly to the case of bounded delays. A low delay bound will cause the node to transmit more frequently than it generates data, in which case many transmission may be empty of data, thus increasing overhead. On the other hand, data aggregation reduces overhead. In fact, if there are no delay bounds, there are no extra transmissions and the effect of aggregation is dominant. We show these effects in Section 5.

In comparing the three strategies above, we expect all three to perform equally well when there are no changes in behavior. The strategy of monitoring the mode may produce false positives. The threshold is critical: too low and there will be many false positives leading to excessive loss of sleep, too high and changes will be detected too late leading to many lost transmissions and corresponding loss of sleep. The effect of changing behavior is described below.

### 4.3.1 Effect of Changing Behavior

An upstream node changes its behavior either by speeding up the rate of transmission or slowing down. Transmission are less likely to be missed if nodes are slowing down, because the receiver wakes up ahead of time. Without adaptation, losses will decrease, but nodes will be staying awake longer than they should be. With mode watching, some sleep will be sacrificed to learn the new behavior, after which nodes will sleep more and lose more (as they originally were). Using sequence numbers should allow the nodes to learn the new behavior in about the same time as the mode watching case learns that the behavior has changed. Thus nodes adapt without sacrificing sleep.

On the other hand, when nodes speed up, the transmission occurs before the receiver wakes up, so losses will increase. This causes sleep to drop as well, because nodes stay awake until the next transmission. Without adaptation, nodes perform very badly, with a significant increase in losses and decrease in sleep. The mode watching case will detect the change and learn the new behavior by staying awake. Again adaptation by sequence numbers will adapt to the new behavior in about the same time as the mode watching case detects it, without sacrificing an entire cycle of sleep.

We use the non-adaptive case as our base for all comparisons. In the next section we describe in more detail the behavior of the base case and determine the range of parameters in which the network should operate for best results.

## 4.4 Impact of Sleep on Benefit

We define the **benefit** of this algorithm as the *factor by which the utility of the network is increased*. The utility, as defined in Section 2, is the number of sensor readings successfully transmitted to the sink from the furthest nodes during the lifetime of the network. A benefit of 1 would mean no increase in utility at all, and a benefit of 10 would mean that 10 times as many readings were delivered to the sink because of the use of the algorithm.

The benefit ( $B$ ) can be expressed in terms of loss ( $L$ ) and sleep ( $S$ ) as:

$$B = \frac{1 - L}{1 - S} \tag{1}$$

Here, we use the loss as seen by the node that has suffered the maximum loss, which is normally the node furthest from the sink. The sleep here is the least in the network, which is usually the node closest to the sink.

To better understand the characteristics of sleep, loss and benefit, we inves-

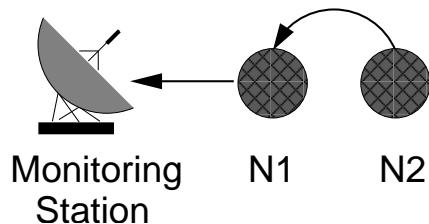


Fig. 4. Simple one-hop network

tigate the case of a simple two-node, one-hop network, as shown in Figure 4. Sleep and loss are a function of several variables, two of which we introduce here:

- $\alpha$  (Sleeping Interval): expressed as a multiple of the upstream node's mean transmission period  $p$ . Every time a transmission is successfully received, the node will sleep for  $\alpha p$  time units, unless it is expecting a transmission from another neighbor. Note that  $\alpha$  can be greater than one.
- $\beta(n)$  (Distribution Cover): a property of the inter-arrival distribution, such that the region  $[1-\beta(n), 1+\beta(n)]$  contains  $n\%$  of the density of the distribution. In the above, the interval is expressed in units of the mean, so it indicates a symmetric interval about the mean. In the rest of the paper we use  $\beta$  to mean  $\beta(99)$ .

Although Algorithm 2 works directly with  $k$  and calculates  $t_k$ , it is useful to consider the general case of each node sleeping for a specific fraction of the period  $p$  and then waking up to receive the next transmission. Figure 5 shows sleep, loss and benefit characteristics as  $\alpha$  varies. The distribution used is a normal distribution and  $\beta$  is 0.1. The sleep shown is from the node closest to the sink (henceforth called  $N_1$ ) and the loss is from the one furthest from the sink ( $N_2$ ). The range of  $\alpha$  can be broken in to several regions:

- $(0, 1-\beta)$ : In this region, there is insignificant loss. Sleep increases linearly and benefit increases sharply.
- $(1-\beta, 1+\beta)$ : Node  $N_1$  is now missing a significant fraction of  $N_2$ 's transmissions. These losses cause the node to stay awake. Soon, sleep begins to decrease because of the amount of time spent awake after missed transmissions, leaving a peak in the sleep and hence the benefit curves. In this region, loss has gone from virtually nil to  $1/2$  - losing every alternate packet. Sleep too is a little more than  $1/2$  - it is sleeping every alternate period.
- $(1+\beta, 2-\beta)$ : In this region, every time the node receives a packet it then goes to sleep for such a long time that it will certainly miss the next one, but catch the one after that. It will continue to miss every alternate packet. The benefit is increasing because sleep is, while loss is practically constant.
- $(2-\beta, 2+\beta)$ : The node may now miss the second transmission too. Thus loss jumps to  $2/3$ . Due to the increase in loss, the sleep, which was increasing, now starts decreasing, leaving another peak, which is higher than the first

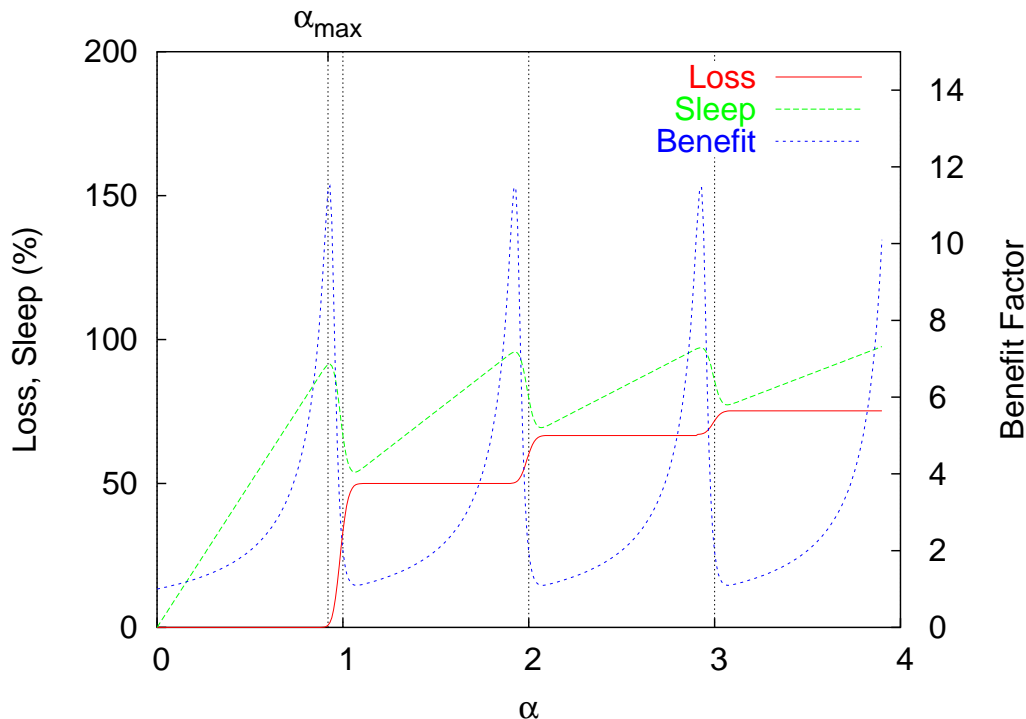


Fig. 5. Sleep, Loss and Benefit

one. The benefit has also peaked. The pattern repeats for the rest of the graph.

Although only a simple scenario is investigated above, the natural intuition is that the alternating and peaky nature of the benefit will hold even in the more realistic case of multi-node, multi-hop networks. With more neighbors, a node will sleep less, but the shape of the curve and the position of the peaks will be retained. A node further away from the sink will lose more, but the stepped nature of the curve and the position of the sharp increases will stay the same. We have verified this intuition with further analysis and simulation, but omit them here for the sake of brevity. We simply report our observation that the net effect on the benefit curve is that the successive peaks of the benefit function decrease in value, instead of all being equal. Thus for practical purposes we need only concentrate on the first benefit peak.

This peak is obtained when  $\alpha = \alpha_{max}$ , which is approximately  $1 - \beta$ . We now focus our efforts on those values of  $\alpha$ . In the next section, we develop a general model for loss, sleep and benefit which can be applied to multi-node multi-hop networks operating in this region.

#### 4.5 Quantitative Model

Of the factors that determine sleep and loss,  $\alpha$  and  $\beta$  are properties of the network and have been introduced above. The other factors are individual to each node and they are:

$d$  (depth): how many hops away from the sink the node is.

$\eta$  (in-degree): how many upstream neighbors the node has. In the case of an uniform density of nodes in the sensor field, the in-degree of a node at depth  $d$  can be obtained by considering the relative number of nodes  $d$  hops and  $d + 1$  hops from the sink, since the former forward traffic for the latter. If  $n$  represents the density of nodes and  $r_{tx}$  the transmission range, then the number of nodes at a distance between  $(d - 1)r_{tx}$  and  $dr_{tx}$  from the base station (which are the nodes  $d$  hops away in the ideal case) is given by  $n\pi\{d^2 - (d - 1)^2\}r_{tx}^2$ . Thus the ratio of the number of nodes  $d + 1$  hops away and the number nodes  $d$  hops away (which is also the average degree of a node at depth  $d$ ) is

$$\eta = \frac{(d + 1)^2 - d^2}{d^2 - (d - 1)^2} = \frac{2d + 1}{2d - 1}. \quad (2)$$

According to this, nodes at depth 1 have the highest degree: three. Very few nodes will have more than that, a fact that that we can use to simplify our calculations.

##### 4.5.1 Loss

Over a single hop, the fraction of transmissions missed depends only on  $\alpha$  and the CDF  $F(\cdot)$  of the upstream node and is equal to:

$$L_1 = \frac{F(\alpha)}{(F(\alpha) + 1)} \quad (3)$$

$$\approx F(\alpha), \text{ since } F(\alpha) \ll 1 \quad (4)$$

In general, the loss at a node at depth  $d$ , which needs  $d$  hops to reach the sink, is given by:

$$L_d = 1 - (1 - L_1)^d \quad (5)$$

$$\approx d * F(\alpha) \quad (6)$$

### 4.5.2 Sleep

For  $\alpha$  in the region of  $\alpha_{max}$ , on average, each node spends a fraction  $(1 - \alpha)$  of the period awake per upstream neighbor when its transmission is not missed. Since  $\eta$  is small, a node with degree  $\eta$  would spend a fraction  $\eta(1 - \alpha)$  awake. The chances of missing a packet are increased by a factor of  $\eta$  to  $\eta F(\alpha)$ . Since a node only sleeps when it has *not* missed a packet, the fraction of sleep is:

$$S_\eta = (1 - \eta F(\alpha))(1 - \eta(1 - \alpha)) \quad (7)$$

### 4.5.3 Benefit

Loss at the outermost tier and sleep at the innermost tier determine the overall benefit. If the network has maximum depth  $d_{max}$  and the degree of depth 1 nodes is  $\eta_1$ , then the benefit is:

$$B = \frac{1 - L_{d_{max}}}{1 - S_{\eta_1}} \quad (8)$$

$$= \frac{1 - d_{max} F(\alpha)}{1 - (1 - \eta_1 F(\alpha))(1 - \eta_1(1 - \alpha))} \quad (9)$$

With  $\beta = 0.1$ ,  $d_{max} = 1$  and  $\eta = 1$ , (9) matches Figure 5. From Equation 9 we have  $\alpha_{max} \approx 0.92$ , which is approximately  $1 - \beta$  as observed in Section 4.4.

## 5 Numerical Results

To verify our model we developed an event-list simulator. The topologies simulated are of a uniform density. The node placement is obtained by first generating a uniform rectangular grid such that nodes placed on that grid will achieve the desired density; then we introduce a small random perturbation in the placement of each node in both the  $x$ - and the  $y$ -direction of the grid. This results in node placements that reflect realistic deployments, generally with uniform density but with some variations that are unavoidable in deployment. Nodes maintain observed interarrival distributions in a 21 bin histogram constructed using the last 400 observations at any point in time, *i.e.* the length of the cycle of observation is 400. Simulations run for at least  $40,000p$  time units, where  $p$  is the largest mean interarrival time over all nodes. A comprehensive set of results is presented in [29]; a representative sample is provided here.

First we show that our intuition about an optimal value of  $\alpha$  (or  $k$ ) that maximizes benefit is correct. To show that benefit is maximum for some value of  $k$ ,

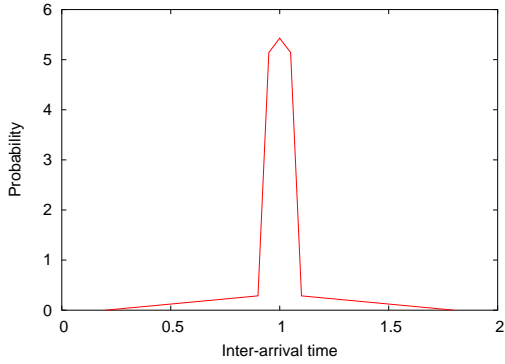


Fig. 6. A Triangular approximation to the normal distribution with a light but long tail

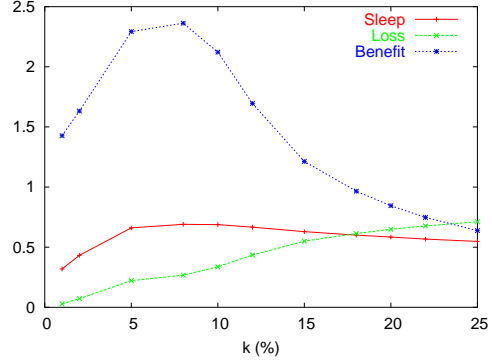


Fig. 7. Loss, Sleep and Benefit for various values of  $k$  for the inter-arrival distribution in Figure 6

we use a triangular approximation to the normal distribution that has a long-but-light-tail as shown in Figure 6 to generate inter-arrival times. We then plot the values of loss, sleep and benefit obtained by simulation for various values of  $k$  in Figure 7. We see that loss increases almost linearly, sleep increases and then decreases as expected. Benefit peaks for  $k \approx 8\%$ . Comparison with Figure 5 shows that it matches the prediction from our model well.

Having shown the need to focus on the small region in which the greatest benefit may be obtained, we perform detailed simulation study of that region. For the rest of the experiments, a truncated normal distribution whose mean ( $\mu$ ) and standard deviation ( $\sigma$ ) are input is used to generate inter-arrival times. The distribution is truncated to  $\pm 3\sigma$  and renormalized. Although this is not a long-but-light-tailed distribution as in Figure 1, it is a close approximation in the region of interest.

In the rest of this section, we systematically present the performance of our algorithm.

### 5.1 Loss Model

Figure 8 shows actual and predicted values of loss against depth for several values of  $k$ . As expected, loss increases almost linearly with depth. However, at higher depths and higher values of  $k$ , the departure from linearity is noticeable. The predicted values are calculated using the more accurate Equation 5 (not its approximation Equation 6). The values for  $F(\alpha)$  are obtained from the loss suffered by the nodes at depth one as in the sleep calculations. The figures show that our model is optimistic, but very close to reality.

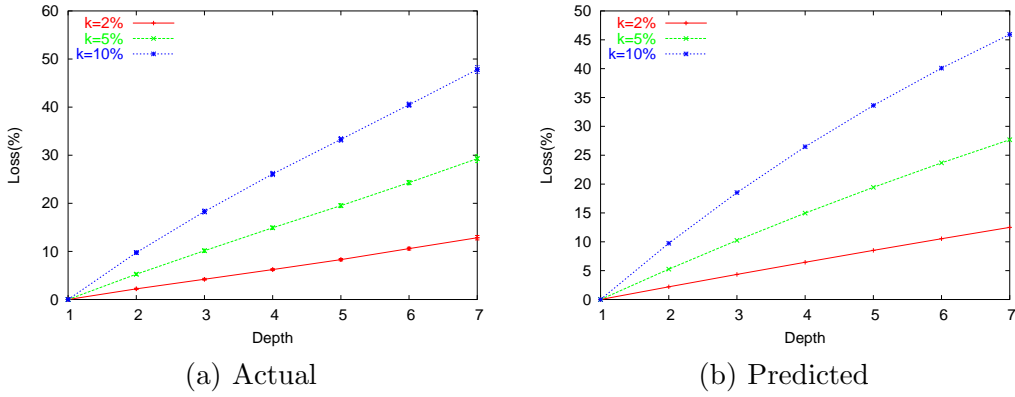


Fig. 8. Effect of depth on Loss, for various  $k$

## 5.2 Sleep Model

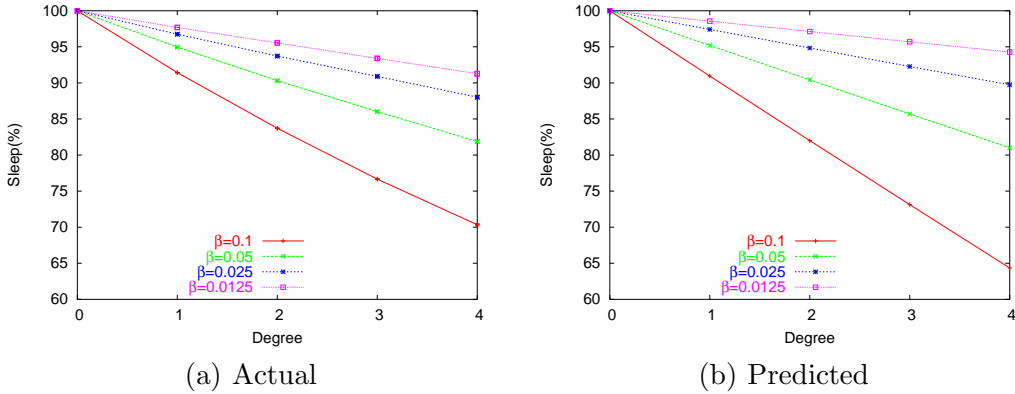


Fig. 9. Effect of in-degree on Sleep for various  $\beta$ ;  $k = 1\%$

To verify our sleep model, we extracted from our simulation runs the sleep attained by each node and its degree under the conditions of unbounded delays and no change in behavior. Figure 9(a) shows sleep against degree of a node for several values of  $\beta$ , both simulated and calculated from the model. As expected, the sleep decreases with increasing nodal degree in an almost linear fashion. The values predicted by the model are shown in Figure 9(b). These values were calculated using Equation 7. The value of  $F(\alpha)$  used in this equation is the average loss suffered by nodes at depth one. Using these,  $\alpha$  was calculated by inverting the distribution being used in the simulation.

We see that the actual values closely match the predicted values for all but the largest values of  $\beta$  and degree. This is because the approximations made in the model are not valid for large values of the degree ( $\eta$ ) and  $\beta$  and the linearity does not hold. We cannot assume that the probability of missing a transmission is proportional to the number of upstream neighbors: it will actually be lesser. The figures show that our sleep model is quite close to

reality, especially for small  $\eta$  and  $\beta$ . Since this is the expected scenario in real networks, the model is useful.

### 5.2.1 Effect of Depth

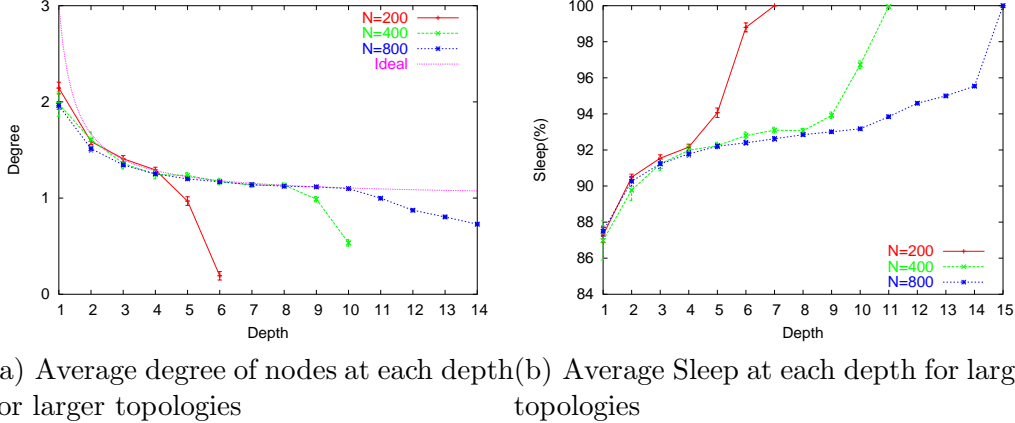


Fig. 10. Relation between degree and sleep in larger topologies

It can be seen that all graphs showing sleep against depth above have a characteristic shape. We show that this is a *characteristic of the topology* and that any sensor network with *uniform distribution of nodes* will have that characteristic. We confirm this by running simulation with larger topologies which have greater maximum depth.

The in-degree of a node i.e. the number of upstream neighbors that a node has determines the sleep fraction of the node. We showed above in (2) that in networks with a uniform density of nodes, the average degree of nodes at a given distance from the sink is  $\eta_d = (2d + 1)/(2d - 1)$ . Figure 10(a) shows average degree at each depth for three topology sizes that we have used. It also shows the ideal value according to Equation 2. We see that all topologies begin in the same way and follow the ideal before dropping abruptly as they reach the fringes of the network. Figure 10(b) shows the average sleep attained by nodes at each depth for the same three topology sizes. Figure 10(b) is approximately an inversion of Figure 10(a) as suggested by Equation 7.

### 5.3 Performance of Adaptation Algorithms

To observe the behavior of the adaptation algorithms, we study the effect of speeding up and slowing down separately as they have different characteristics. We also limit the change to nodes at a specific depth, so that the effect stands out. We then consider a general case where some nodes slow down and

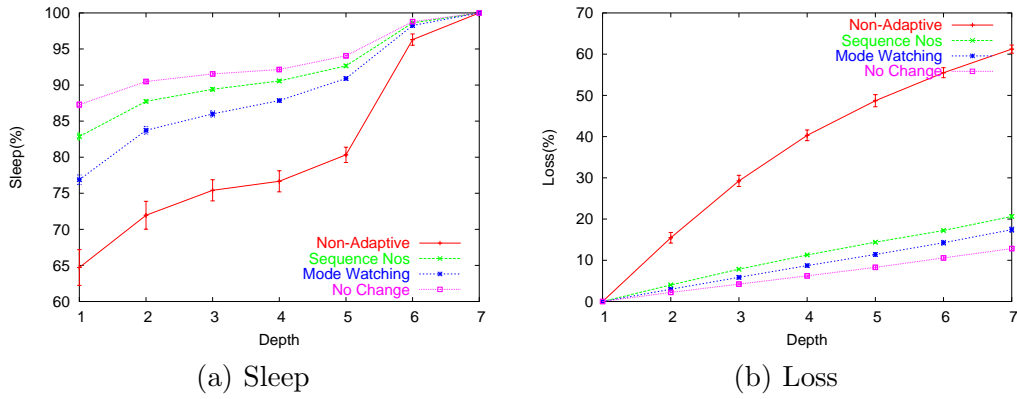


Fig. 11. Effect of Speeding up: all algorithms

some speed up. For ease of labelling, the performance of adaptation without sequence numbers is labelled with the legend “mode watching”. For comparison, we also study the performance of the basic non-adaptive algorithm under the same conditions. Also included in each figure is the performance of the non-adaptive algorithm when nodes do not change behavior - this provides the basis for comparing the performance of the adaptation algorithms.

### 5.3.1 Effect of Speeding up

To compare the performance of the adaptation algorithms when some nodes are speeding up, we run simulations where 10% of nodes in the network (chosen at random) decrease their mean transmission period by 10% over a cycle. Figure 11(a) shows sleep values for the non-adaptive, sequence number and mode-watching algorithms. Also shown is the case when no change occurs (with the non-adaptive algorithm) for comparison. Figure 11(b) shows the corresponding loss values. We see that the non-adaptive case performs the worst: it loses a lot of sleep and also loses much more data than the other algorithms. Adaptation by observing the histogram works better. It is able to detect the changes and adapt. In the process, it loses some sleep - this is the cost of staying awake to learn the new behavior. The sequence number adaptation case works best. It learns the new behavior without sacrificing sleep.

To better understand the reaction of each algorithm to speeding up, we investigate each algorithm separately, limiting the speedup to nodes at a particular depth each time.

#### Non-Adaptive Case

First, we investigate the effect of speeding up on the non-adaptive algorithm. To see the effect better, the change in transmission period is limited to nodes

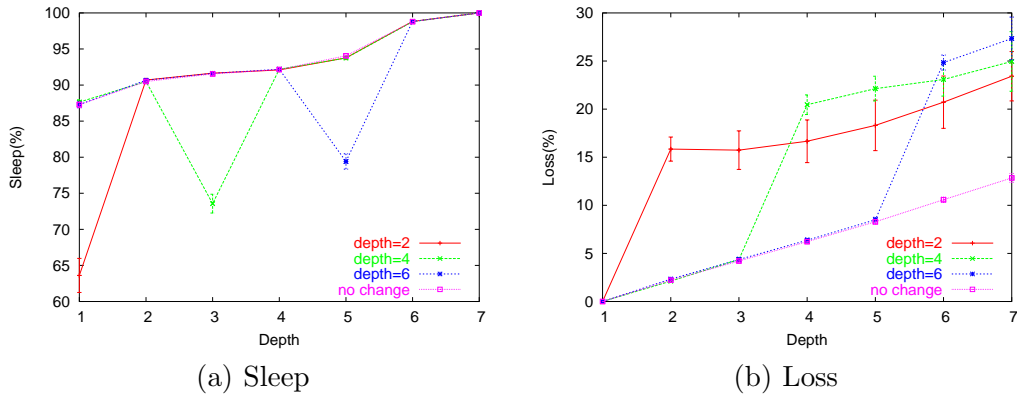


Fig. 12. Effect of Speeding up: non-adaptive case at three depths.

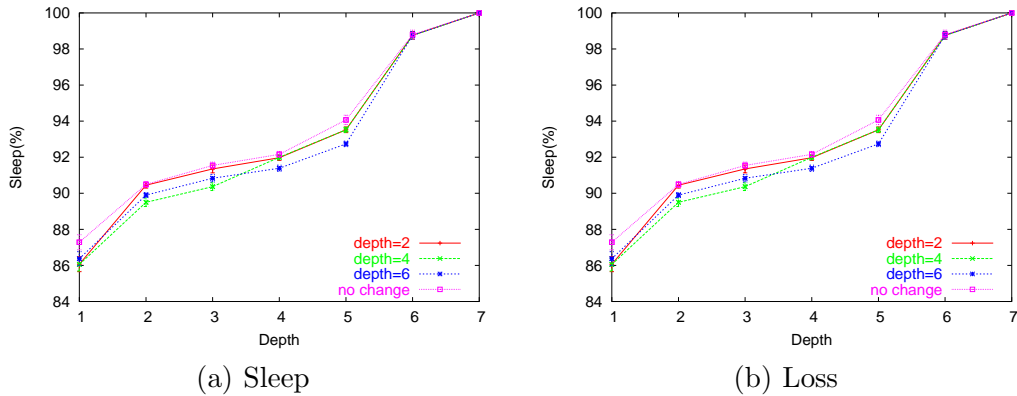


Fig. 13. Effect of Speeding up: Adaptation with Sequence Numbers, at three depths.

of a particular depth alone. We run simulations with the depth at which the change occurs set to 2, 4 and 6 separately. Figure 12(a) shows the effect on sleep and Figure 12(b) shows the effect on loss. In the loss graph, we see an increase in the loss at the depth at which the change occurs and beyond it. In the sleep graph, we see a drop one hop upstream from the depth at which the change occurs. Both effects are to be expected: since transmissions are arriving before they are expected, they are missed and the upstream nodes stay awake waiting for the next one. We also see that the sleep of nodes upstream from the ones that are changing is not affected. The loss, however is increased. This too is expected: their data passes through nodes at the depth at which change occurs, where it may be lost.

### *Adaptation with Sequence Numbers*

Figure 13(a) shows how sleep is improved by using sequence numbers to detect and adapt to changes. Again, changes occur at depths 2, 4 and 6. Figure 13(b) shows the corresponding loss values.

We notice a difference in the way sleep and loss changes in this case compared

to the non-adaptive case. In the non-adaptive case, sleep and loss at depths downstream to the affected depth are unaffected by the change, whereas sleep decreases and loss increased at downstream nodes for the adaptive case. This is true of the other adaptive algorithm too as shown below. This is because the non-adaptive algorithm does not change its traffic shape in response to changes it observed. On the other hand, the adaptive algorithms lowers the transmission period when its sees that an upstream neighbor has lowered its. Thus the effect of a change propagates towards the sink in the adaptive cases, whereas it does not in the non-adaptive case.

### Adaptation without Sequence Numbers

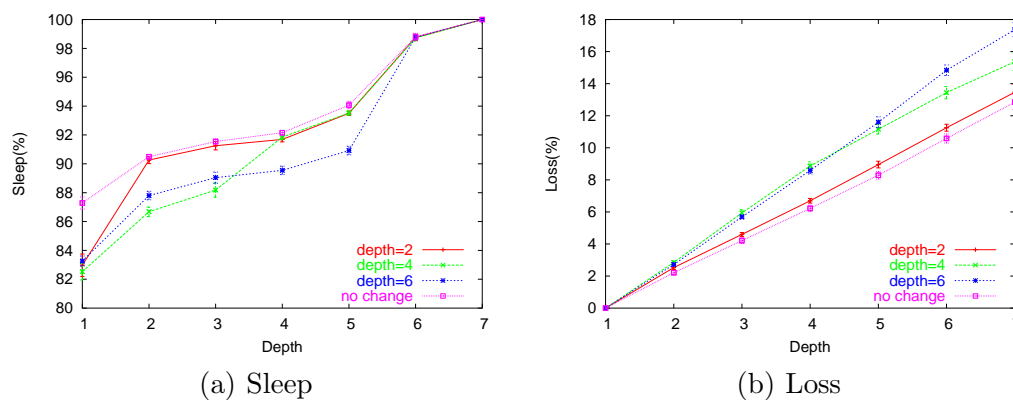


Fig. 14. Effect of Speeding up: adaptation by mode watching, at three depths

We expect the mode watching adaptive method to perform similar to the sequence number method, but losing more sleep. This intuition is confirmed by Figure 14.

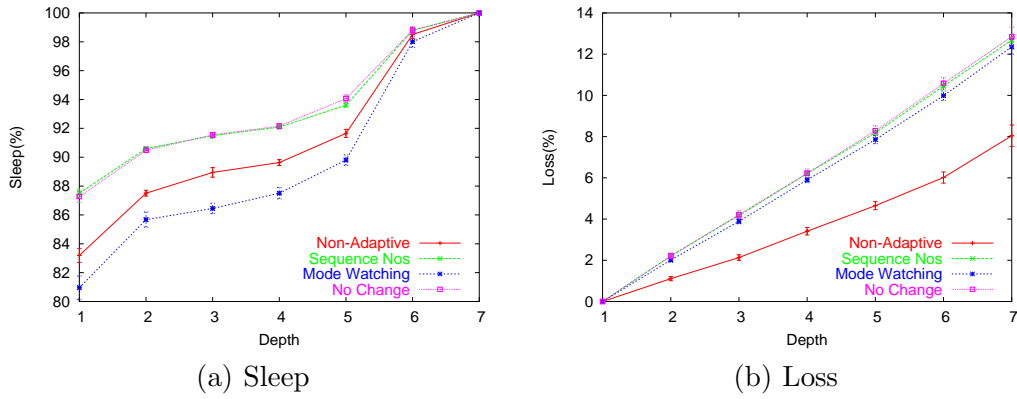


Fig. 15. Effect of Slowing down: all algorithms

### 5.3.2 Effect of Slowing down

To compare the performance of the adaptation algorithms when some nodes are slowing down, we run simulations where 10% of nodes in the network (chosen at random) increase their mean transmission period by 10% over a cycle. Thus the change is gradual not abrupt. Figure 15(a) shows sleep values for the non-adaptive, sequence number and mode-watching algorithms. Also shown is the case when no change occurs (with the non-adaptive algorithm) for comparison. Figure 15(b) shows the corresponding loss values. We see that the non-adaptive case performs the worst: it loses a lot of sleep although it loses less data than the other algorithms. The sequence number adaptation algorithm works better. It is able to detect that the period has increased and that it can sleep longer. Thus sleep increase, while loss decreases only slightly because the algorithm is soon sleeping more, which increases the chances to loss.

The mode observing adaptation algorithm does not perform so well. It does detect that the period has increased, however, by staying awake to learn the new behavior it is sacrificing a lot of sleep. There is no increase in loss due to this type of change and the sleep lost can significantly cut into the benefit. Over a very long period of time, this sacrifice may pay off because of the sleep gained by learning the new behavior. This is under the unlikely assumption that the nodes that slowed down will settle into the new behavior and not change again.

To better understand the reaction of each algorithm to slowing down, we investigate each algorithm separately, limiting the slowdown to nodes at a particular depth each time.

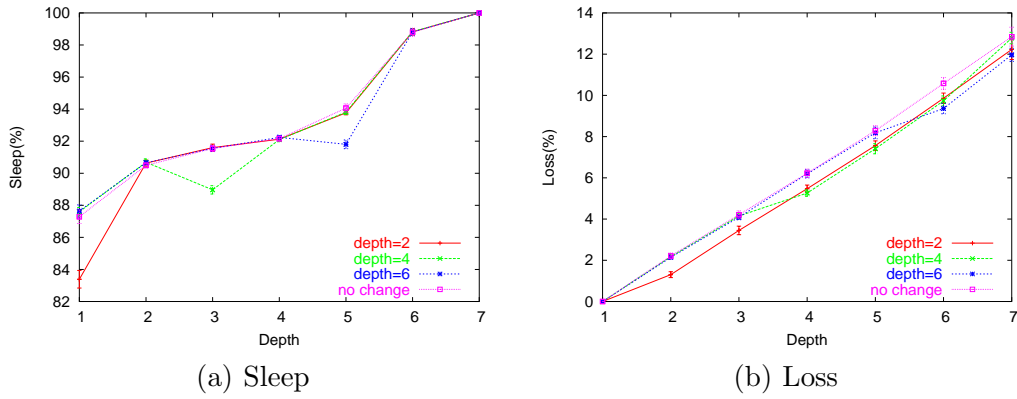


Fig. 16. Effect of Slowing Down: non-adaptive case at three depths.

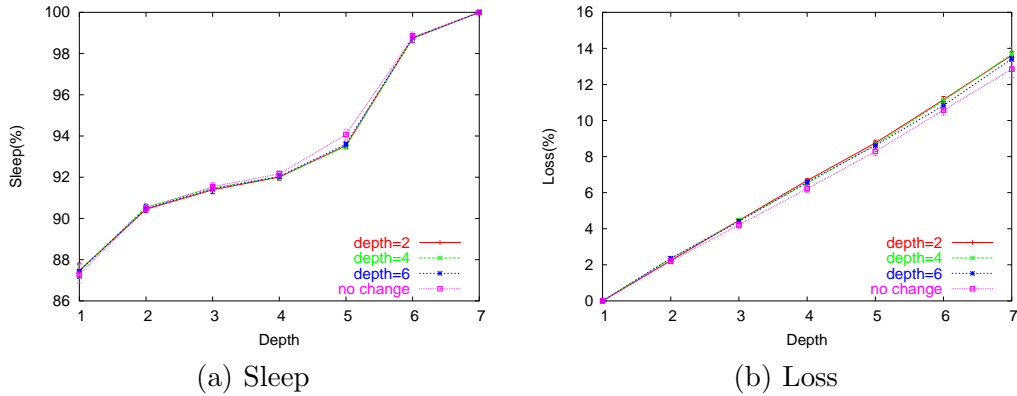


Fig. 17. Effect of Slowing Down: Adaptation using Sequence Numbers at three depths.

### *Non-Adaptive Algorithm*

To see the performance of the non-adaptive algorithm when nodes slow down, we limit the slowdown to one depth and run simulations with depths 2, 4 and 6. Figure 16(a) shows the effect on sleep and Figure 16(b) shows the effect on loss. In the loss graph, we see a decrease in the loss at the depth at which the change occurs and beyond it. In the sleep graph, we see a drop one hop upstream from the depth at which the change occurs. Both effects are to be expected: since transmissions are mostly arriving after they are expected, there are fewer misses. However, the upstream nodes sleep less because they are waking up early.

We also see that the sleep of nodes upstream from the ones that are changing is not affected. The loss, however is decreased. This too is expected: their data passes through nodes at the depth at which change occurs, where it less likely to be lost.

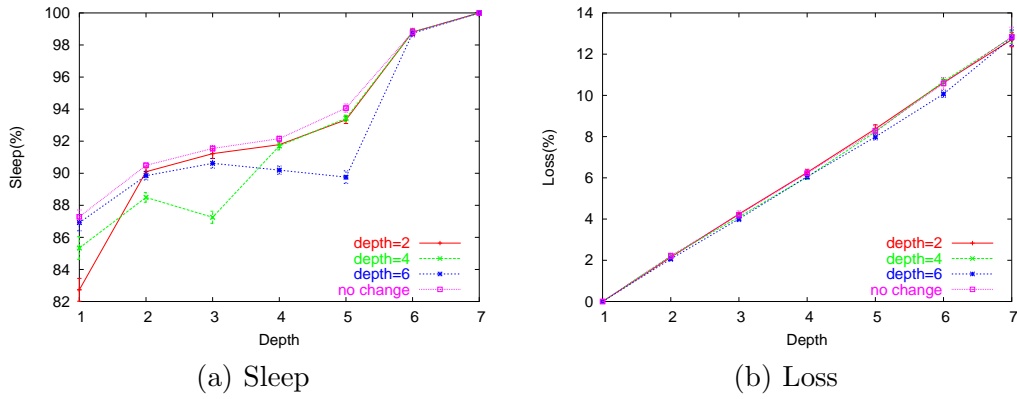


Fig. 18. Effect of Slowing Down: Adaptation without sequence numbers at three depths.

### *Adaptation using Sequence Numbers*

Figure 17 shows that adaptation by sequence numbers performs better than the non-adaptive case. It is able to detect that the inter-arrival times are longer and starts sleeping more too. Although it loses less when the change starts to happen, it loses as much as before after it has adapted to the new period. This explains why the sequence number adaptive algorithm loses slightly more than the non-adaptive case. In terms of overall benefit it is still the superior algorithm as the savings in sleep more than make up for the slightly greater loss. In fact, over a longer period of time this benefit will be even greater.

### *Adaptation without Sequence Numbers*

As mentioned at the beginning of the section, the observe-and-react algorithm does not perform as well as the others. In other respects, the performance is similar to the sequence number based algorithm. This is shown in Figure 18. If we consider a very long time frame in which sleep sacrificed by this algorithm is negligible, then it can be said to be as effective the sequence number adaptation. Realistically, we expect that nodes will drift many times in their lifetime and their immediate downstream neighbor will lose a significant portion of its sleep if they use this algorithm. This is especially true for nodes closest to the center, as they will have more upstream neighbors than nodes further away.

### *5.3.3 Speeding Up and Slowing Down*

We consider the more realistic case where some nodes speed up and some slow down. In the simulations, change in the period is limited to  $\pm 10\%$  in either direction. Figure 19 shows sleep and loss in such a scenario for the non-adaptive case and the two adaptation algorithms. Included is the “no-change” case for comparison.

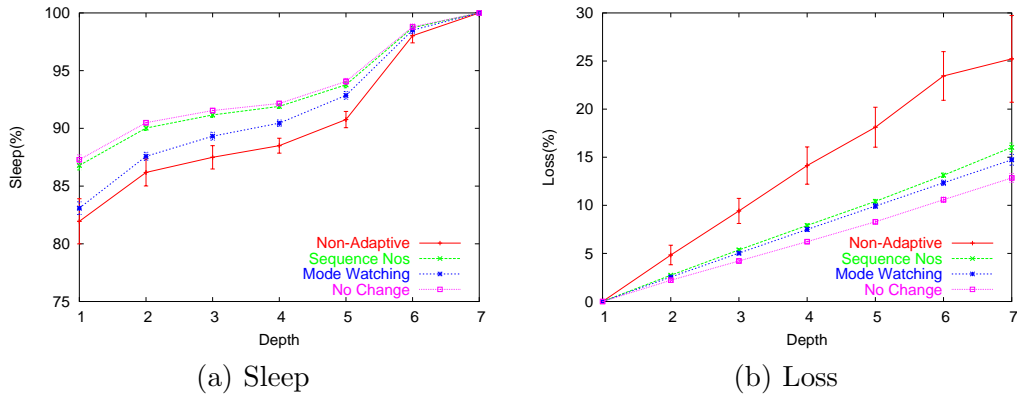


Fig. 19. Effect of Change: Adaptation Algorithms Compared

We see that adaptation using sequence numbers performs the best, being very close to the ideal “no-change” values. Without adaptation a large penalty is observed in both sleep and loss. The second adaptation algorithm loses more sleep because it stays awake to learn the new behavior when it detects a change. It loses less because it does not lose anything in the periods in which it stays awake. In general, some increase in loss is seen for both as expected.

#### 5.4 Bounded Delay

Finally, we present results of experiments when nodes are allowed to specify bounds on the delay before the traffic originated by them reaches the monitoring station. In Section S:algorithm we stated that our algorithm can operate taking such delay bounds into consideration, and here we present the performance of our algorithm.

Figures 20-23 show the results for a scenario in which every node specifies a delay bound equal to its sensing period. The delay bound forces a smaller transmission period (about 1/7 of the unbounded period in our simulations), while the spread of the inter-arrival distribution remains the same. Effectively, we have a larger value of  $\beta$  and hence a smaller  $\alpha$ . This leads to lower losses and sleep in the bounded delay case. We see that the overall delay is much less in the case of bounded delays as required. The tradeoff is the overhead: there are significantly more transmissions with delay bounds – these are the dummy transmissions.

Figure 23 presents results related to the number of transmissions. The quantities plotted are the number of extra transmissions that nodes at various depths have to make *due to aggregating and shaping traffic periodically*, as described in Section 3. Consider first the curve for unbounded delay case. As expected, the aggregation that is performed in order to shape the traffic results in lowering the number of transmissions (“Extra” Tx is negative) at all depths except

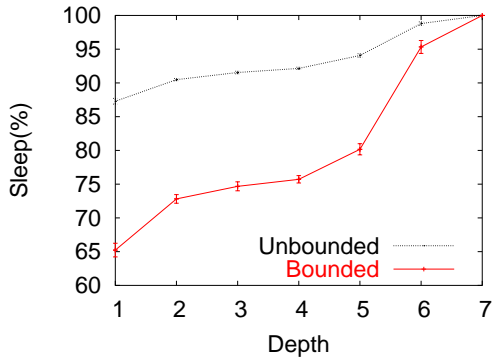


Fig. 20. Bounded Delay: Sleep

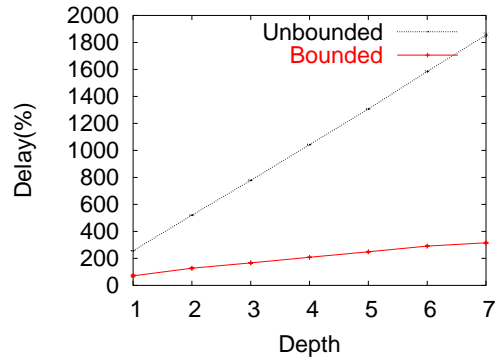


Fig. 22. Bounded Delay: Delay

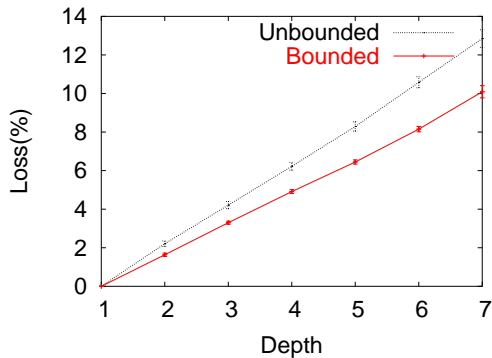


Fig. 21. Bounded Delay: Loss

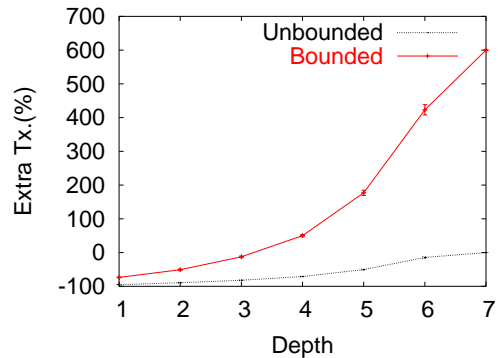


Fig. 23. Bounded Delay: Extra Tx.

the last one. Naturally, aggregation does not make any difference to the outermost nodes (which have no incoming transmissions to forward or aggregate). However, at inner nodes, there is reduction; in general nodes of less depth have to forward more traffic and thus gain more from aggregation.

When delay bounds are introduced, dummy transmissions come into play and form a counter-effect to this reduction, by *increasing* the number of transmissions required. This effect is more pronounced at higher depths, because delay bounds must be divided over a larger number of hops, requiring smaller transmission periods. However, the tradeoff is favorable. As we can see in Figure 23, the number of transmissions remains *lower* than the case with no shaping in the first few hops. In particular, in the crucial first hop nodes (where the sleep is minimum), the number of transmissions remains almost the same. This means that the advantage of aggregation, which allows the inner nodes to sleep more, is present in the case of bounded as well as unbounded delays.

## 6 Conclusion

We have investigated the problem of designing a self-organized scheduling algorithm to allow nodes in a wireless sensor network to switch off their wireless

transceivers when not needed in order to increase network utility. We articulated how the amount of sleep affects the utility function. Our algorithm adapts to slow changes of the transmission characteristics of nodes. By simulation, we verified the behavior of the network in the crucial region of high utility. Our algorithm is very robust and provides good benefit.

We are working on several continuing aspects of this research. One area of further study concerns the loss rate of sensor readings from nodes of different depth, or distance from the monitoring station. The loss is higher from nodes of different depths, and the loss from the deepest nodes governs the performance, similarly the sleep at the nodes closest to the monitoring station. We are working on techniques to “flatten” these characteristics to improve the benefit of the algorithm. A testbed implementation on Berkeley motes is also under way.

## References

- [1] I. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, A survey on sensor networks, *IEEE Communication Magazine* 40 (8) (2002) 102–116.
- [2] I. Chlamtac, M. Conti, J.-N. Liu, Mobile ad hoc networking: imperatives and challenges, *Ad Hoc Networks* 1 (2003) 13–64.
- [3] R. Ramanathan, R. Hain, Topology control of multihop wireless networks using transmit power adjustment, in: *INFOCOM* (2), 2000, pp. 404–413.  
URL [citeseer.nj.nec.com/ramanathan00topology.html](http://citeseer.nj.nec.com/ramanathan00topology.html)
- [4] R. Wattenhofer, L. Li, P. Bahl, Y.-M. Wang, Distributed topology control for power efficient operation in multihop wireless ad hoc networks, in: *Proc. of INFOCOM*, 2001.
- [5] S. Narayanaswamy, V. Kawadia, R. S. Sreenivas, P. R. Kumar, Power control in ad-hoc networks: Theory, architecture, algorithm and implementation of the COMPOW protocol, in: *Proc. of European Wireless 2002. Next Generation Wireless Networks: Technologies, Protocols, Services and Applications*, Florence, Italy, 2002, pp. 156–162.  
URL [http://black.csl.uiuc.edu/~prkumar/ps\\_files/compow\\_ewc\\_2002.pdf](http://black.csl.uiuc.edu/~prkumar/ps_files/compow_ewc_2002.pdf)
- [6] W. Ye, J. Heidemann, D. Estrin, An energy-efficient MAC protocol for wireless sensor networks, in: *Proceedings of the IEEE Infocom, USC/Information Sciences Institute, IEEE, New York, NY, USA, 2002*, pp. 1567–1576.  
URL <http://www.isi.edu/johnh/PAPERS/Ye02a.html>
- [7] S. Singh, C. Raghavendra, Power efficient MAC protocol for multihop radio networks, in: *The Ninth IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, 1998, pp. 153–157.

- [8] R. Zheng, J. Hou, L. Sha, Asynchronous wakeup for ad hoc networks, in: Proceedings of the 4th ACM international symposium on Mobile ad hoc networking and computing(MOBIHOC), ACM Press, 2003, pp. 35–45.
- [9] T. van Dam, K. Langendoen, An adaptive energy-efficient mac protocol for wireless sensor networks, in: Proceedings of the first international conference on Embedded networked sensor systems(SENSYS), ACM Press, 2003, pp. 171–180.
- [10] R. Rozovsky, P. R. Kumar, SEEDEx: A MAC protocol for ad hoc networks, in: Proc. of The ACM Symposium on Mobile Ad Hoc Networking & Computing, (MobiHoc), Long Beach,CA, 2001, pp. 67–75.
- [11] Y.-C. Tseng, C.-S. Hsu, T.-Y. Hsieh, Power-saving protocols for ieee 802.11-based multi-hop ad hoc networks, in: Proc. of INFOCOM, Vol. 1, 2002, pp. 200–209.
- [12] M. Stemm, R. H. Katz, Measuring and reducing energy consumption of network interfaces in hand-held devices, IEICE Transactions on Communications E80-B (8) (1997) 1125–1131.
- [13] L. M. Feeney, M. Nilsson, Investigating the energy consumption of a wireless network interface in an ad hoc networking environment, in: Proc. of IEEE INFOCOM, 2001, pp. 1548–1557.
- [14] D. E. Culler, J. Hill, P. Buonadonna, R. Szewczyk, A. Woo, A network-centric approach to embedded software for tiny devices, in: Proc. of the First International Workshop on Embedded Software (EMSOFT), 2001.
- [15] S. Singh, C. Raghavendra, PAMAS: Power aware multi-access protocol with signalling for ad hoc networks, ACM Computer Communications Review. URL [citeseer.nj.nec.com/460902.html](http://citeseer.nj.nec.com/460902.html)
- [16] E.-S. Jung, N. Vaidya, A power saving mac protocol for wireless networks, Technical Report, UIUC (Jul. 2002).
- [17] S. Singh, M. Woo, C. S. Raghavendra, Power-aware routing in mobile ad hoc networks, in: Mobile Computing and Networking, 1998, pp. 181–190. URL [citeseer.nj.nec.com/singh98poweraware.html](http://citeseer.nj.nec.com/singh98poweraware.html)
- [18] M. W. Subbarao, Dynamic power-conscious routing for MANETS: an initial approach, in: IEEE Vehicular Technology Conference, 1999, pp. 1232–1237.
- [19] T. A. ElBatt, S. V. Krishnamurthy, D. Connors, S. Dao, Power management for throughput enhancement in wireless ad-hoc networks, in: IEEE International Conference on Communications, 2000, pp. 1506–1513.
- [20] J. P. Monks, V. Bhargavan, W.-M. W. Hwu, A power controlled multiple access protocol for wireless packet networks, in: Proceedings of INFOCOM, 2001, pp. 219–228.
- [21] J.-C. Cano, P. Manzoni, Evaluating the energy-consumption reduction in a manet by dynamically switching-off network interfaces, in: Proc. of the Sixth IEEE Symposium on Computers and Communications, 2001, pp. 186–191.

- [22] R. Kravets, K. Schwan, K. Calvert, Power-aware communication for mobile computers, in: Proc. of the International Workshop on Mobile Multimedia Communications(MoMuc'99), 1999.  
URL [citeseer.nj.nec.com/kravets99poweraware.html](http://citeseer.nj.nec.com/kravets99poweraware.html)
- [23] B. Chen, K. Jamieson, H. Balakrishnan, R. Morris, Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks, in: MOBICOM 2001, 2001.
- [24] R. Min, M. Bhardwaj, N. Ickes, A. Wang, A. Chandrakasan, The hardware and the network: Total-system strategies for power aware wireless microsensors, in: Proc. of the IEEE CAS Workshop on Wireless Communications and Networking, Pasadena, CA, USA, 2002.  
URL <http://www.mit.edu/~rmin/research/min-cas02.pdf>
- [25] E. Shih, S.-H. Cho, N. Ickes, R. Min, A. Sinha, A. Wang, A. Chandrakasan, Physical layer driven protocol and algorithm design for energy-efficient wireless sensor networks, in: Proc of Mobicom 2001, 2001.
- [26] M. L. Sichitiu, Cross-layer scheduling for power efficiency in wireless sensor networks, in: Proc. of Infocom 2004, 2004.
- [27] K. S. Trivedi, Probability and Statistics with Reliability, Queuing, and Computer Science Applications, John Wiley and Sons, New York, 2001.
- [28] B. R. Haverkort, R. Marie, G. Rubino, K. Trivedi, Performability Modelling, Wiley, Chichester, England, 2001.
- [29] S. Visweswara, An ad-hoc adaptive power conservation scheme for sensor networks using switch-off, Master's thesis, North Carolina State University, publicly available via WWW: <http://www.lib.ncsu.edu/ETD-db/> (2004).