Abstract

VISWESWARA, SHARAT C. An Automatic, Adaptive, Ad-hoc Algorithm for Power Conservation in Sensor Networks using Switch-off (Under the guidance of Dr. Rudra Dutta).

Wireless sensor networks are set to revolutionize the way we sense and control our environment. A set of nodes equipped with sensors and capable of communicating with each other in ad-hoc fashion, they relay their data to a monitoring station. What sets them apart from other ad-hoc networks is the periodic nature of the traffic and a critical dependence on a finite source of power, the battery. The battery is a bottleneck to the lifetime of the network and careful use of the battery can increase the lifetime by orders of magnitude.

The periodic nature of traffic in a sensor network can be leveraged to switch the transceiver off when not needed, as shown in existing literature. We take up one such switch-off-switch-on scheduling algorithm and show how it can be extended to be robust to dynamic conditions. We then develop an analytical model which can be used to predict the performance of the algorithm. We also identify a better way of expressing the utility of the network and quantify the benefit accruing from using the algorithm, making it possible to compare different variations of the algorithm and different configuration parameters. The model can also be used to find ways to improve the performance of the algorithm to obtain greater power efficiency.

An Automatic, Adaptive, Ad-hoc Algorithm for Power Conservation in Sensor Networks using Switch-off

by

Sharat C. Visweswara

a thesis submitted to the graduate faculty of

north carolina state university

in partial fulfillment of the

requirements for the degree of

master of science

department of computer science

raleigh

July 2004

To my parents and my sister...

Biography

Sharat C. Visweswara is a graduate student in the Department of Computer Science at the North Carolina State University. He was born and brought up in Bangalore, India. In 2001 he received a Bachelor of Engineering degree in Computer Science and Engineering from Bangalore University. After working breifly in Bangalore, he was admitted to the North Carolina State University in 2002. He is currently working towards his Master of Science degree in Computer Science.

Acknowledgements

I would like to thank my advisor Dr. Rudra Dutta at NCSU for guiding me as I worked on this thesis. I have thoroughly enjoyed working with him and have learnt a lot from him.

I thank Dr. Mladen Vouk and Dr. Mihail Sichitiu for serving on my advisory committe and providing valuable advice. I am further thankful to Dr. Sichitiu for providing the code used to generate network topologies. I also thank Apurva Goel for writing parts of the simulator code.

My family and my friends have been very supportive; without them this work would not have been possible. I am very grateful to them for their love and support.

The research in this thesis was supported in part by National Science Foundation grant EEC-033-2271.

Table of Contents

List of Figures

Chapter 1

Introduction

Computer communication networks have made tremendous advances in the last few years. The need for providing connectivity to people and devices on the move has led to the *ad hoc* networking paradigm. Ad hoc networks are self-organizing, self-configuring set of nodes, with a minimum of a-priori assignment of network roles such as routing. Wireless sensor networks are a class of ad-hoc networks meant to sense and control our environment. These networks are usually composed to small nodes with the ability to sense some characteristic of the environment and the ability to communicate wirelessly with neighboring nodes.

Sensor nodes are usually distributed randomly in the environment when deployed. There is a monitoring station nearby which sinks all the data from the sensor nodes. The nodes must organize themselves; establish a topology and routes such that each node generates data and may also forward data from other nodes towards the monitoring station. In a sensor network topology can change over time. However, the primary cause is not node mobility as in the case in general ad-hoc networks. Topology changes in a sensor network because nodes consume their batteries and die or new nodes are added to the network.

Sensor nodes may be equipped with a finite power source: a battery. When a node's battery runs out, it is rendered useless. When too many nodes have died, the network as a whole is rendered practically useless because it is unable to perform its primary function of sensing the environment anymore. Also, the loss of a few critical nodes (those closest to the monitoring station) can cut off the sensor network from the outside world, thus no data can be obtained form the network. In fact, since the nodes closest to the monitoring station bear must to the maximum amount of *forwarding* traffic, they will be the first to die a phenomena termed the doughnut effect in [26]. Sometimes sensor networks are deployed in hostile conditions and it may be difficult or impossible to replace or recharge batteries when they run out. In such cases, it is essential that the battery be used sparingly so as to maximize the operating time of the network.

One way to minimize battery use is to switch off components of the node when they are not needed. The transceiver is a major consumer of power and significant savings are possible if the transceiver is switched off when not needed. *Idle listening* wastes power because the transceiver is neither sending nor receiving. Sensor nodes typically spend a lot of time with their transceivers in the idle-listening mode and hence elimination of idle-listening has been a major area of research.

Eliminating idle-listening presents problems of its own. Since there is no out-of-band notification to a node that a transmission to it is imminent, the only way to know when a transmission is coming in is to listen for it. Thus by switching off its transceiver, a node risks missing transmissions. If it cannot afford to miss transmissions, the node must know in advance when the next transmission is expected, so that it can wake up in time.

The idea of scheduling transmissions such that the destination node can switch off its transceivers when not needed and knows ahead of time when to wake up has been explored before. One such scheduling algorithm is presented in [10]. In our work, we extend that approach to adapt to dynamic conditions. We also develop a model to predict the performance of the algorithm on a given network. We also quantify the benefit to a network of using the algorithm and determine the conditions that result in maximum benefit.

The rest of the thesis is structured as follows. In Chapter 2 we survey the field of wireless ad-hoc and sensor networks, particularly power conservation. In Chapter 3 we summarize the prior work that our work is based on and define in terms of that work the issues we are addressing. In Chapter 4, we describe the algorithms we have developed to allow nodes to adapt to dynamic conditions. In Chapter 5 we first describe a metric for expressing the utility of a network and the advantage of the algorithm. Using this, we determine the conditions under which benefit is maximized and develop expressions for the characteristics of the scheduling under these conditions. Results of our simulations are presented in Chapter 6. In the last chapter, we draw conclusions about our work and discuss possible future work in this direction.

Chapter 2

Context

Packet radio networks were first developed in the 1970's [33, 31], since then they have grown tremendously. The wireless networks of today can be broadly classified into two categories. The first category is that of *infrastructure* networks like the wireless LAN, which take centralized approach. Recent standards like IEEE 802.11 and Bluetooth have made these types of networks very popular. The other category is *ad hoc* networks, also called *instant infrastructure* networks. In such networks, all participants are peers and there is no centralized control. Ad hoc networks are self-organizing and self-configuring, requiring no external help.

The nature of ad hoc networks presents unique challenges. Mobility in an important characteristic of most application of ad hoc networks. Thus traditional routing and quality of service approaches are insufficient, for example. The same mobility introduces the problem of power conservation as mobile devices are usually dependent on a small, finite power supply. It also present the problem of *locating* the devices. The many aspects of ad hoc networks are surveyed in [6], some of which are presented in this chapter.

Wireless sensor networks are a specialization of ad hoc networks; they are meant for the specific purpose of sensing the environment in which they are deployed. They differ from the general class of ad hoc networks in several ways. Mobility is usually not a characteristic of sensor networks. However, topologies are dynamic: this is caused by a high failure rate of the nodes, and deployment of new nodes to replace the failed ones. Power conservation and scalability assume a greater importance in the case of wireless sensor networks. The latter part of this chapter discusses some of the issues regarding wireless sensor networks.

A comprehensive survey is presented in [2].

2.1 Ad Hoc Networks

The first ad hoc networking applications were the DARPA Packet Radio Network project (PRNet) in 1972, followed by the Survivable Radio Networks project (SURAN) in 1983 which improved over PRNet [6, 13]. As technology improved, packet networks became more feasible and useful [9]. Typically, the most important ad hoc deployments have been military. Since the mid 1990's, with the definitions of standards such as IEEE 802.11, commercial applications have been emerging rapidly.

2.1.1 Challenges

Mobile ad hoc networks (MANETs), present some unique challenges. MANETs are formed dynamically by independent, mobile nodes without the use of a centralized facility such as a base station. Nodes may move and arrange themselves randomly, which make the topology highly dynamic and unpredictable. Data from one node may have to be forwarded by several other peer nodes before reaching the destination. Thus ad hoc networks work in a *multi-hop* fashion and every node is a router.

With no infrastructure required, ad hoc networks make it very convenient to set up a network. Once deployed and activated, nodes find each other and organize themselves. However, this convenience and flexibility comes at a price. In addition to their unique constraints, MANETs inherit the problems of wireless networking:

- the medium is unprotected; external signals can interfere with communication
- the wireless medium is more prone is errors
- propagation characteristics of the medium are non-uniform, asymmetric and time-varying.
- hidden terminal and exposed terminal problems occur.

Characteristics, complexities and constraints specific to ad hoc networks are [7, 6]:

- Lack of infrastructure: since each node operates in distributed peer-to-peer mode, network management, such as routing, must be distributed across all nodes. This makes the nodes more complex.

- Dynamic topologies: because nodes can move arbitrarily, the network topology can change randomly and rapidly. Adjusting transmission and reception parameters such as power can also impact the topology. This can result in packet loss and even network partitions.
- Variation in node capabilities: heterogeneity in node radio capability leads to asymmetric links. Processing and battery capability may also vary.
- Energy constraints: since nodes use limited batteries, it limits the capability of the node. For example, a node with a small battery should occur on few routes between other nodes as possible as forwarding requires additional energy.
- Scalability: traditional Internet protocols and algorithms were designed to work on fixed networks. Wireless networks lack many of the characteristic of wired networks, scuh as heirarchy, making these algorithms and protocols unsuitable for wireless networks.

2.1.2 Routing

The highly dynamic nature of ad hoc network topologies makes routing a challenging task. Numerous routing algorithms have been proposed; some of the important ones are surveyed in Ref. [22], whereas a more comprehensive survey of routing protocols is presented in Ref. [20]. Routing protocols can be classified according to the type of *cast*, i.e. whether the nodes use a *Unicast, Broadcast, Multicast* or *Geocast* forwarding [6].

With broadcasting, messages are received by all who sense the transmission. Routing by broadcasting is done by *flooding*, i.e. every nodes forwards every messages it receives. This ensures all nodes in the network receive the message, one of which will be the destination. Flooding is highly redundant and is very inefficient for most applications. It must be done intelligently to avoid redundancy and increase efficiency.

Multicast routing protocols are needed when a node needs to send the same message to several destinations. In that sense, geocast routing is a special case of multicast routing that is used to deliver messages to all nodes in a specified geographical area. Nodes may join or leave a multicast group as desired, on the other hand, nodes can join or leave a geocast group only by entering or leaving the corresponding geographical region.

Most routing protocols use the unicast principle; one source sends to one destination.

These protocols may be *table-driven* or *demand-driven* [22].

Table-Driven Routing Protocols: Table-driven protocols maintain current, up-to-date routing information from each node to every other node in one or more tables. To ensure consistency of routing information updates are sent to all nodes in the network when the topology changes. The individual protocols in this category differ in the types of routing related information in the tables and the methods by which nodes are informed about changes. Destination-Sequenced Distance-Vector Routing (DSDV) [18], Clusterhead Gateway Switch routing (CGSR) [5], the Wireless Routing Protocol (WRP) [15] are table driven protocols.

Demand-Driven Routing Protocols: This type of routing only creates routes when they are needed. When a node requires a route to a destination it initiates a route discovery process within the network. The process is complete when a route is found or all possible route permutations have been examined. Once a route is established, it is maintained by a route maintenance procedure until the destination is inaccessible or until the route is no longer needed. Ad Hoc On-Demand Distance Vector Routing (AODV) [19], Dynamic Source Routing (DSR) [11], Temporally Ordered Routing Algorithm (TORA) [17] are demand-driven protocols.

2.1.3 Power Conservation

Mobile devices use a battery as their energy source. Batteries are finite and improvements in battery technology are not expected to be increase their capacities significantly in the near future. In these circumstances it is essential that power utilization is efficient. Ways must be found to minimize the use of the battery, across all components of the mobile device. For example, schemes have been proposed at the level of the operating system, such as CPU scheduling. In mobile devices, it has been found that networking activities account for a significant fraction of power consumed, as much as 50% [12]. Thus it is possible to significantly decrease power utilization by managing networking activities efficiently.

An important characteristic of wireless network interfaces is that power consumed in the idle, receiving and transmitting states of the same order of magnitude, whereas the power consumed in the *sleep* state is an order of magnitude lesser. Power consumed by the wireless network interface can be reduced by using network protocols and algorithm that maximize the time spent by the interface in the *sleep* or *off* state by minimizing time spent in the idle state. These are *switch-off* approaches to saving power, and our work falls in this category. There are other approaches that attempt to conserve energy without switching off, which we describe below.

Non Switch-off Approaches

Non switch-off strategies are usually *global* in nature: they have the network as a whole in mind, not individual nodes. Several network-level approaches fall in this category and usually involve routing. Power aware routing protocols take one of two approaches [6]: minimum-energy or maximum-lifetime. Minimum-energy routings minimize the energy consumed to forward a pocket from source to destination, whereas the other approach selects routes and transmission power levels to achieve optimal lifetime [21].

A balance between minimum-energy and maximum-lifetime is presented in Ref. [32], in which four different route selection schemes are proposed. **Minimum Total Transmission Power Routing** (MTPR) attempts to minimize the total transmission power required to forward a packet from source to destination. Over a single hop, the power required to transmit successfully depends on the distance to the node, error rate of the channel and any interfering noise. MTPR selects the path that minimizes the total required transmission power over all hops, but by doing so, it may choose paths with more hops to the destination. Such routes are unstable, as the chances of failure are greater. A better approach is to consider reception power in addition to transmission power, this will tend to select shorter routes. MTPR suffers from the drawback that it may over-utilize a single node, i.e. it is not fair. **Minimum Battery Cost Routing** (MBCR) corrects this by including remaining battery life at each node on the path in its calculations of cost. This way, it avoids over-using a single node. However, it may still select paths in which nodes have little battery left when it is possible to select a path with a higher cost, but with more battery power remaining and thus attain a longer network lifetime. To correct this **Min-Max Battery Cost Routing** (MMBCR) is proposed, which balances battery cost and the lifetime of each node and thus the network. Although this is fairer, it still does not guarantee that minimum total transmission power path will always be chosen. The **Conditional Min-Max Battery Cost Routing** (CMMBCR) algorithm is proposed as an improvement. In this scheme, when all nodes in some possible routes between a source and a destination have sufficient remaining battery capacity (i.e. above a threshold), a route with minimum total transmission power is chosen. However if all routes have nodes with low battery capacity (below threshold), routes including nodes with the lowest battery capacity are avoided to extend the lifetime of these nodes.

The **Common Power Protocol** (COMPOW) [16] has an additional goal: to reduce contention at the MAC layer. Reducing transmission power and hence the range of the device not only saves power directly, it also saves power by reducing the chances of collisions. With fewer collisions, less power is wasted in failed transmissions. This also has the effect of increasing the traffic capacity of the network. COMPOW attempts to find the minimum common power level at which the network is still connected. The common power level ensures the bi-directionality of links. To find the minimum common power level, COMPOW maintains multiple routing tables, one for each transmit power level available. Once this table has been formed, the optimal power level is the smallest power level whose routing table has the same number of entries as the routing table would have without the COMPOW protocol. The approach described in [30] is similar.

Switch-off Approaches

As discussed at the beginning of this section, communication equipment in mobile devices draw a significant amount of power [29]; of which the idle and receive states consume a major part [8]. Thus energy is wasted when the node is waiting for a transmission. Also a node wastes energy in receiving transmission that are not destined for it. Thus switching off communication equipment when the node is not involved in any communication is an attractive proposal.

These savings come at a cost; there are many trade-offs that are to be kept in mind. Switching the transceiver on and off may require more energy than idling; in which case too many on-off transitions in a short period of time may significantly reduce power savings [4]. When the transceiver is switched off, the node cannot receive any transmissions. There is no out-of-band notification of an incoming message. Thus there is a possibility that transmissions will be missed. The node will remain unaware of these losses unless there is a mechanism is detect them and inform the node. These issues can be overcome and there are many different switch-off approaches which operate at the MAC layer.

For example, a node can switch off its transceiver when it hears a conversation between two other nodes. Some approaches like [4] take advantage of the RTS/CTS signalling mechanism of the MAC layer to know when there will be conversation between two other nodes. The node can capture the header of the transmission that follows the RTS/CTS handshake and know the length of the transmission, from which it can determine when to wake up. PAMAS [27] extends this by using a separate channel for RTS/CTS signalling. SEEDEX [23] takes a different approach: each node *schedules* transmission randomly and communicates its schedule to other nodes by means of the seed it is using for its random number generator.

2.2 Wireless Sensor Networks

In recent times, advances in wireless communication technology, digital electronics and sensing technology has made it possible to build small, low-cost, low-power sensors with wireless networking and computing capability. *Wireless sensor networks* are a large number of such devices deployed around the phenomena to be observed.

The position of sensor nodes need not be fixed. They can be distributed randomly, say by sprinkling from a plane. This requires sensor networks to have the self-organizing ability of ad hoc networks; in fact wireless sensor networks are a class of ad hoc networks. However, sensor networks differ from ad hoc networks in many ways, which are described in the next section. There has been significant interest in this area recently; Ref. [1] surveys the many areas currently being investigated.

2.2.1 Challenges

Sensor networks inherit most of the challenges of ad hoc networks, since the share the same characteristics such as multihop communication and dynamic topologies. They also differ from ad hoc networks in the following ways [20, 1]:

- Sensor networks are typically an order of magnitude larger than ad hoc networks in terms of the number of nodes. Thus it is essential that algorithms for routing, etc., scale well. It also makes it difficult to simulate sensor networks.

- Sensor nodes are prone to failure; they are cheap and densely deployed to counter this. However, the application and algorithms must be robust to such failures. This causes the topology to change frequently.
- Sensor nodes are usually not moved once deployed. In such a circumstance, mobility ceases to be a factor that causes topology changes. Topology can still change due to nodes dying or being added.
- Sensor node usually use a broadcast communication paradigm, whereas general ad hoc networks use unicast.
- Sensor nodes are limited in power, computational capacity and memory. In fact sensor networks are critically sensitive to battery power efficiency, more than general ad hoc networks.
- Sensor nodes may not have a global ID.
- Sensor networks usually have an *egress* traffic pattern with relatively low volume and high regularity.

Some of these turn out to be advantages such as the low mobility and predictable traffic pattern. Others, such as the limited memory, computing ability and critical sensitivity to battery power efficiency add to the challenges of designing algorithms for sensor networks. Power conservation in sensor networks has been an important topic of research, as this is seen to be the largest factor holding back the utility of sensor networks.

2.2.2 Power Conservation

Many of approaches suitable for ad hoc networks are based on the assumption that the battery is expensive, but not limited, i.e. it can be recharged or replaced. For the most part, this is not true for sensor networks: many sensor networks are deployed in hostile environments where it is not possible to replace or recharge batteries. In addition, unlike typical MAC level power saving algorithms for ad hoc networks, energy conservation and self-configuration are primary goals, fairness and latency are less important [35].

Approaches similar to those described in Section 2.1.3 have been applied to sensor networks. The high density of sensor nodes are an advantage when using transmit power adjustment to conserve power [16, 21, 34]. Others work at the MAC layer to turn off the transceiver when not needed[28, 35]. Yet others focus on routing[3], taking advantage of the low mobility.

Integrated approaches

All approaches described above focus on one layer of the networking stack. Although these can be combined to obtain better energy conservation than the individual ones, integrated approaches that take into account all layers and components of the network in the design are likely to be the best. In [14], for example, collaboration across the hardware, algorithms and networking stack improves energy-efficiency and extends operational lifetime. The techniques include dynamic voltage scaling, distributed processing, and power aware routing, combined with a careful characterization of the quality of the channel and an API that allows selection of appropriate transmission power. A physical layer based approach is described in [24] shows how to design physical layer aware protocols, algorithms, and applications that minimize energy consumption of the system. Their approach prescribes methods that can be used at all levels of the hierarchy to take advantage of the underlying hardware.

A third alternative is to leverage the predictability of the traffic pattern in sensor networks. Since it is known that nodes generate and forward data at regular intervals, it is possible to deduce when a node will be communicating and when it will be idle with some confidence. One such integrated *scheduling* approach is described in [25]. In this approach, nodes communicate with each other in an initial setup phase to cooperatively agree on a schedule of transmissions and receptions. This enables the node to sleep most of the time and wake up only when necessary to transmit its data or forward data from other nodes. A similar approach, but one in which nodes automatically learn the information they need by observing their neighbors and does not require ideal periodicity is studied in Ref. [10]. This approach is robust to uncertainty in the periodicity of transmissions. It is assumed that the time interval between successive transmissions varies according to some probability distribution; such traffic is referred to as *quasi-periodic*. In contrast to using a communicated schedule, nodes learn the information they need by observing their neighbors and adjusting their behavior as neccessary.

2.3 Our Contribution

Our work extends the automatic approach described above [10]. We extend the algorithm to make it robust to changes in the periodicity of transmissions. We then show that a purely temporal definition of network lifetime does not suffice under these circumstance and must be broadened. We also develop a model that allows us to predict the performance of the algorithm, given the characteristics of wireless sensor network on which it is operating.

Chapter 3

Problem Definition

The problem of conserving power by switching off is treated in [10]. It is shown that quasi-periodicity necessitates traffic shaping. Unless a node shapes its own traffic and makes it quasi-periodic, its downstream neighbor will get very little sleep. Assuming traffic is shaped by each node, the author shows how nodes can learn the behavior of their upstream neighbors and determine a sleep schedule such that they do not lose more transmissions than is specified as a parameter. For completeness, we present a summary of the core algorithms described in [10].

3.1 Prior Work

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.

The 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.

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.

3.1.1 Representation of Traffic Shape

The term *traffic shape* refers to mean the probability distribution of inter-arrival times. The probability distribution is represented as a histogram. 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.

3.1.2 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, its 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

Algorithm 1 Calculate Transmission Period

```
{$n.d_{max}$: 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
$t_{tx} \leftarrow min\{u.t_{tx}\} \forall u \in U$
$self.t_{tx} \leftarrow min\{d_{max}, t_{tx}\}$
```

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 calculates 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.

3.2 Our Contribution

The shape of traffic originating from a node may change over time. *Reshaping* may occur for many reasons. Some changes are *unintentional*, such as clock drift. In such cases, the node does not know that its behavior is changing. Other changes are *intentional*, in which case the node knows that its behavior will change. This may be an application requirement, say a command from the monitoring station to generate data more or less frequently than it is currently. A *notification* approach would handle intentional cases better: the node which is required to change its behavior would explicitly notify its downstream neighbor that it will change its behavior. The downstream neighbor would then react appropriately.

Algorithm 2 Calculate Sleep Time

```
{$n.S$: node $n$'s traffic shape, treated as a c.d.f}
{$n.last$: node $n$'s last arrival time}
{$k$: allowable loss factor}
{$U$: set of all upstream neighbors}
{$now$: current time}
$min \leftarrow \infty$
for each $u$ in $U$ do
  find $t_k \mid u.S(t_k) = k$
  $ETA = u.last + t_k$
  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 $sleep$
```

3.2.1 Adaptation

In our work, we address the case of unintentional reshaping and describe how it can be detected and adapted to. This requires active monitoring by the observer as the transmitting node has no knowledge of this change is thus of no help here.

By *reshaping*, we mean that the inter-arrival distribution changes in one of two ways: the mean of the distribution either increases or decreases. This corresponds to nodes slowing down and speeding up their transmission rates respectively. We assume that the variance of the distribution does not change. This is a reasonable assumption as the variance is determined by network characteristics, such as node density, network congestion, the MAC layer, etc. - factors that are not under the control of the node.

3.2.2 Model

We also develop an analytical model of the sleep and loss characteristics exhibited by the basic algorithm when traffic shapes are stable and do not change. We identify the parameters that determine these characteristics and express sleep and loss in terms of these parameters. This allows us to predict the performance of the algorithm given the conditions in which it is operating.

3.2.3 Network Lifetime

We also show that under the conditions of *qausi-periodicity*, where there is to some extend a trade-off between loss and sleep, a simple definition of network lifetime is not sufficient to quantify the advantage of the algorithm. We introduce the terms *utility* and *benefit* to describe the advantage of using the algorithm. To quantify the advantage we relate benefit to sleep and loss.

Chapter 4

Adaptation Algorithms

We now address the problem of adapting to reshaping. As mentioned in 3.2.1 the inter-arrival distribution changes in one of two ways: the mean of the distribution either increases or decreases. This corresponds to nodes slowing down and speeding up their transmission rates respectively.

A node cannot rely on observations made in the third phase of the algorithm. If a transmission is missed and the next one caught, *the observed inter-arrival time is very large*. Thus missed transmissions cause spuriously long inter-arrival times to be observed, which distort the histogram if used in it. Hence, before a node can detect and adapt to reshaping, *it must account for distortions caused by missed transmissions.*

We present two algorithms to accomplish this. For comparison, we also discuss the *non-adaptive* strategy articulated in [10]. All three may be considered specializations of the third phase of the basic algorithm. We also discuss the effect of reshaping with these algorithms, considering the cases of slowing down and speeding up separately.

4.1 The Current Non-Adaptive Algorithm

In this approach, the node neither observes nor adapts. It knows the 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 shape after the node starts sleeping. If the behavior of upstream neighbors 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. On the other hand, in the event of reshaping, this approach fails.

Consider the case of a node whose upstream neighbor slows down, i.e. the mean of its inter-arrival distribution increases. The node wakes up ahead of time, which means fewer transmissions will be lost. The node will spend more time awake, which translates to lesser sleep. The amount of sleep lost is proportional to the increase in the mean; the greater the increase the greater the loss of sleep.

Consider the case of a node whose upstream neighbor speeds up, i.e. the mean of the inter-arrival distribution decreases. The node now wakes up later than it should, which means more transmissions are lost. Since the algorithm is to stay awake until a transmission is received, lost transmissions lead to lost sleep because the node will now stay awake until the next transmission, which will on average be one period away.

Thus we expect that sleep is decreased by any type of movement in the shape of the traffic from the node being observed. Waking up too early or too late both lead to decresed sleep: this suggests that there is a point at which sleep is maximised. We show that this is indeed true and determine this optimum point in Section 5.1.

4.2 Adaptation Using Sequence Numbers

One possibility is to actively identify missed transmissions and account for them. To aid in this, we use sequence numbers. Every node numbers its transmissions successively. Its downstream neighbor will also keep track of the last sequence number seen from it, or alternatively the next one expected. If the downstream node receives a transmission with a sequence number greater than that expected from the transmmitting node, it will know exactly how many transmissions it has missed.

The observed inter-arrival time, which will be very large in such a case, can then be divided equally to obtain the successive inter-arrival times. They can be adjusted, if necessary, to ensure that the missed transmission(s) arrived before the node woke up from its sleep. The successive inter-arrival times calculated this way will not be the actual inter-arrival times, but they are sufficiently accurate that the values can be used in the histogram without distorting it. Hence the histogram remains valid and usable at all times.

Using this modification, the node can continue to record observations and use them in the third phase of the algorithm. This means that the node can continuously adapt to changes in upstream neighbor behavior.

When an upstream node slows down, i.e. the period increases, the node will initially be waking up too early and sleeping less as in the non-adaptive case above. However, the node will soon know the new shape of the traffic because it is recording the inter-arrival times. It will adapt to this shape and will increase the amount of time it sleeps.

On the other hand, when an upstream node speeds up, i.e. the period decreases, the node will initially be waking up too late as in the non-adaptive case. It will be missing more transmissions and thus sleeping less. However, it knows how many it has missed because of the sequence numbers in the packets. It will calculate apprximate inter-arrival times of all the transmissions it missed and the one it received and use it in the histogram. When the histogram reflects the new shape, the node will have adapted and know it must wake up earlier than it did. It will now lose less and therefore sleep more.

Thus we expect that by using sequence numbers, nodes can adapt quickly to reshaping.

4.3 Adaptation Without Sequence Numbers

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.

The *mode* of the histogram is a property that will not be changed by the distortion caused by sleeping. In [10] it is shown that the mode will change only if the behavior of the upstream node changes. Thus the node may monitor the mode of the histogram and if it changes by too much, the node concludes that the neighbor being observed has changed it behavior. The node reacts by switching to phase one of the algorithm. It will stay awake all the time and try to learn the new behavior from scratch, before attempting to sleep again.

When an upstream node slows down, the mean of its inter-arrival time distribution increases and so does the mode. About a cycle after the change happens, the receiving (downstream) node will observe the shift in the mode. Similarly when a node speeds up, its downstream neighbor will notice the mode shift to the left in its histogram. In either case, it will stop sleeping and switch to phase one of the algorithm. When it enters phase three it will again sleep.

Thus it is possible to adapt to reshaping without using sequence numbers. However, we do not expect it to perform as well as adaptation with sequence numbers for two reasons. These reasons also provide an insight into some of the parameters of the algorithm that determine utility and hence are worth noting. Firstly, a lot of sleep is lost because the node detecting the change stops sleeping. This reduces the lifetime of the network and thus its utility. Secondly, because of the ambiguity in determining when the mode has shifted, we expect false postives. Lifetime is further reduced because of this.

Two factors determine the possibility of false positives: the aqccuracy of the shape and the change threshold. The more accurate the shape and the higher the threshold, the lesser the chances of false positives. We note that the algorithm's notion of the shape is derived from a histogram. This cannot be expected to be accurate; *it will always be an approximation.* Thus the mode of the histogram can vary about its mean position even when there is no change in behavior. A larger number of observations, i.e. a larger *cycle* would provide a more accurate shape, but the algorithm would then take proportionally longer to detect changes. Once it has detected a change, it would stay awake longer too. Hence the cycle should be just long enough to provide a *sufficiently accurate* idea of the shape. Anything more is counter-productive.

There is a simular argument for the threshold. If set too low, it will be very sensitive to changes and cause more false positives, reducing the network utiliy. If set too high, it may fail to detect some changes and thus reduce network utility too.

An improvement can be made if it is assumed that the largest possible single inter-arrival time is lesser than two successive shortest possible inter-arrival times. In other words, if a transmission is missed, then the observed inter-arrival time should always be greater than that observed for a successful transmission. At be beginning of phase three of the algorithm, the node can *freeze* the histogram bounds, i.e. record the minimum and maximum values already in the histogram and refuse to use any values that fall outside that range. This will ensure that the large inter-arrival times caused by missed transmission will never be entered into the histogram. Additionally *this is an easy way to detect a missed transmission.* If more transmissions than expected are being missed, then the node would conclude that the upstream neighbor has changed its behavior and react as described above. The restrictive assumtions may not be valid for real-world inter-arrival distributions, although they may be valid for truncated distributions used in simulation studies (Chapter 6). Hence these improvements are not investigated.

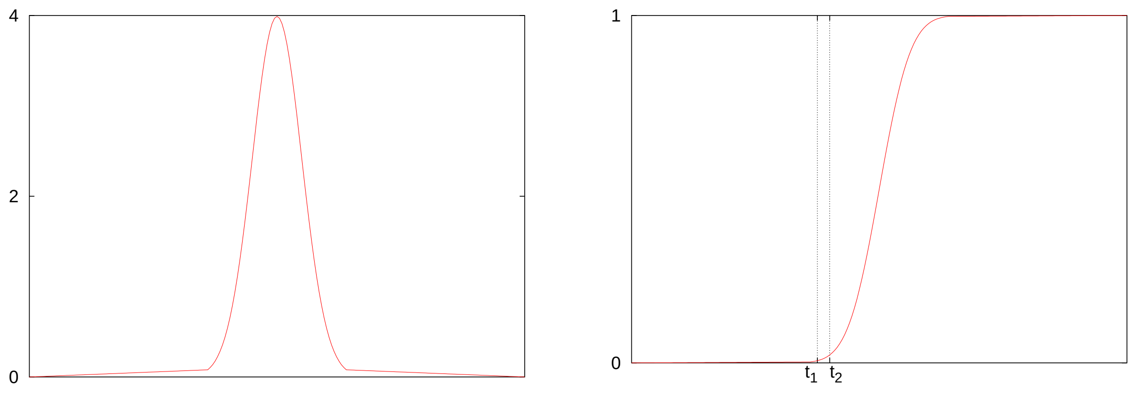
Chapter 5

Model of Basic Algorithm

The previous chapter describes several variations to the basic algorithm. Also, we show in Section 4.1 that the same algorithm can behave better or worse depending on its parameters. To be able to compare these configurations with each other, we need a measure of the advantage offered by a particular configuration. This leads us to define the terms *utility* and *benefit* below.

5.1 Utility and Benefit

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 (Figure 5.1(a)) we have assumed, 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 should really 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).



(a) PDF (b) CDF

Figure 5.1: Light-but-long-tailed inter-arrival distribution

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.

The *benefit* of an algorithm is the *factor by which the utility of the network is increased by using the algorithm*. A benefit of 1 would mean no increase in utility at all, a benefit of 10 would mean that 10 times as many readings were delivered to the sink because of the use of the algorithm and so on.

On a single node, benefit is directly expressible in terms of the fraction of time spent sleeping and the fraction of data lost by that node. From the definition, we express benefit (B) in terms of Sleep (S) and Loss (L) as:

$$B = \frac{1 - L}{1 - S} \tag{5.1}$$

Over an entire network, however, benefit cannot be determined in terms of a single node's sleep and loss. As shown in [10] and explained in Section 5.3 the outermost nodes lose the most data and the innermost nodes sleep the least. The lifetime of the network is determined by the lifetime of the innermost nodes; when they die the network is disconnected

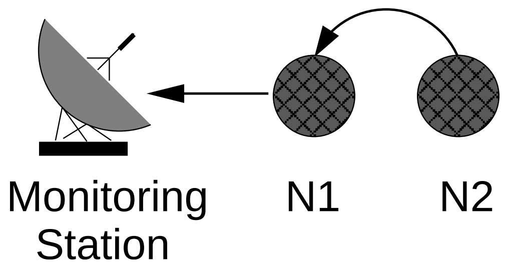


Figure 5.2: A simple two-node, two-hop network

and useless even if all the other nodes are alive. Hence benefit over a network should be related to the fraction of time spent asleep by those nodes one hop away from the base. There is more freedom in choosing the source of the loss metric; we settle on the average loss suffered by the nodes furthest from the base.

To better understand the characteristics of sleep, loss and benefit, we investigate the case of a simple two-node, two-hop network.

5.2 Model of a simple two-node network

The network is as shown in Figure 5.2. The nodes are N_1 and N_2, and N_2 is sending data to N_1 to be forwarded to the sink. Since N_2 is not going to change its behavior, N_1 uses the *non-adaptive* algorithm described in Section 4.1. Using the guidelines in the preceeding paragraph, we seek to understand the behavior of sleep at N_1, loss at N_2 and benefit over the whole network.

We use the following terms and quantities in our mode:

- p (Period): the mean inter-arrival arrival time from the upstream neighbor.
- F: the c.d.f of the inter-arrival distribution, but in terms of a multiple of p, i.e., the mean is always 1.
- α (Sleeping Interval): expressed as a multiple of the upstream mean period p. Every time a transmission is successfully received, the node will sleep for αp time units. Note that α can be greater than one.

  The relation between α and t_k is straightforward:

$$\alpha = \frac{t_k}{p} \tag{5.2}$$

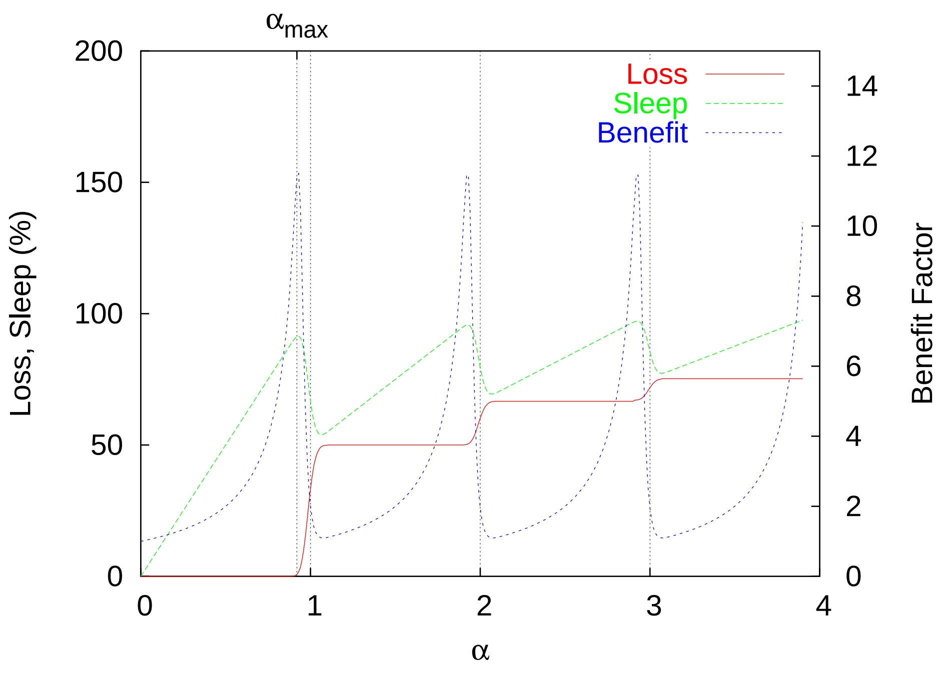


Figure 5.3: Sleep, Loss and Benefit

- $\beta(n)$ (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. From this point onwards, we use β 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 5.3 shows sleep, loss and benefit characteristics as α varies. The distribution used is a normal distribution and β is 0.1. The sleep shown is from the node closest to the sink (N_1) and the loss is from the one furthest from the sink (N_2).

The range of α 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 half: losing every alternate packet. Sleep too is a little more than half: 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 one. The benefit has also peaked. The pattern repeats for the rest of the graph.

Although only a simple scenario is investigated above, the alternating and peaky nature of the benefit holds even in the case of more realistic 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. In Section 5.3 we present a more general model of sleep, loss and benefit.

From Figure 5.3, we see that the maximum benefit is obtained when $\alpha = \alpha_{max}$, which is appromimately $1-\beta$. In other words, to obtain maximum benefit from the algorithm, it must be run with values of k such that $t_k \approx p\alpha_{max}$. In this range, k and $F(\alpha)$ are very small.

We now focus our efforts on this range of α. 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.

5.3 General Model of Loss, Sleep and Benefit

In the general case, two more properties of a node are needed to determine sleep and loss:

- d (depth): the number of hops to the sink.
- η (in-degree): how many upstream neighbors the node has. In the case of an uniform distribution of nodes in the sensor field, the average degree of a node

at depth d is: $\eta = (2d+1)/(2d-1)$. 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.

5.3.1 Loss

Over a single hop, the fraction of transmissions missed depends only on α and the distribution and is equal to:

$$\begin{aligned} L_1 &= \frac{F(\alpha)}{(F(\alpha)+1)} \qquad (5.3) \\ &\approx F(\alpha)\,,\text{ since } F(\alpha) \ll 1 \qquad (5.4) \end{aligned}$$

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

$$\begin{aligned} L_d &= 1-(1-L_1)^{(d-1)} \qquad (5.5) \\ &\approx (d-1)*F(\alpha) \qquad (5.6) \end{aligned}$$

5.3.2 Sleep

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

$$S_\eta = (1-\eta F(\alpha))(1-\eta(1-\alpha)) \qquad (5.7)$$

Note that is the degree is very large, we cannot make the assumption that the probability of missing a packet in a period are multiplied by the degree. Since we have already shown that η will rarely be greater than three, we can make this assumption and simplify our calculations.

5.3.3 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 η_1, then the benefit is:

$$\begin{aligned} B &= \frac{1 - L_{d_{max}}}{1 - S_{\eta_1}} && (5.8) \\ &= \frac{1 - (d_{max} - 1)F(\alpha)}{1 - (1 - \eta_1 F(\alpha))(1 - \eta_1(1 - \alpha))} && (5.9) \end{aligned}$$

With $\beta = 0.1, d_{max} = 1$ and $\eta = 1$, (5.9) matches Figure 5.3. From Equation 5.9 we have $\alpha_{max} \approx 0.92$, which is approximately $1 - \beta$ as observed in Section 5.2.

Chapter 6

Simulation Results

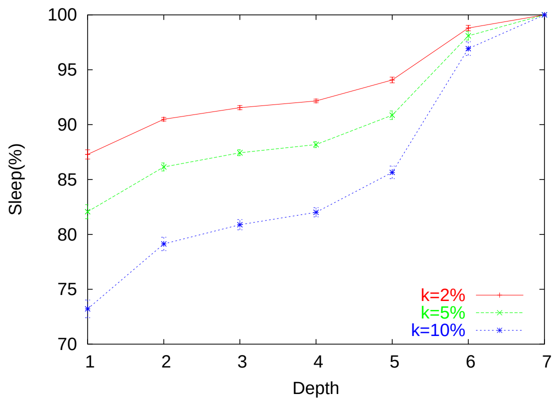
We have develop a custom, event-list simulator to verify and validate our hypotheses. The sensor network topologies used have uniform density, and nodes are placed in a grid with some perturbation. Unless otherwise indicated 200 node topologies with a maximum depth of 7 are used. Inter-arrival times are generated using a bounded normal distribution whose mean p and standard deviation σ are input. The distribution is truncated at $\pm 3\sigma$ and re-normalized. Although it is not a long-but-light-tailed distribution like Figure 5.1(a), it a sufficiently close apprximation in the region of interest. Nodes maintain observed inter-arrival distributions in a 21 bin histogram with a cycle size of 400 i.e. the histogram is constructed using the last 400 observations at any point in time. Simulations run for at least $40,000p$ time units. Unless otherwise mentioned, $k = 2\%$, $\beta = 0.05$ and delays are unbounded.

The fraction of time spent by each node sleeping is calculated from after the time the node enters phase three of the algorithm. In the initial phases, the node is always awake and this time is not included in calculating the fraction of time spent sleeping. Loss is calculated as the fraction of readings from a node that fail to reach the sink. Both sleep and loss are averaged for each depth and plotted against the depth.

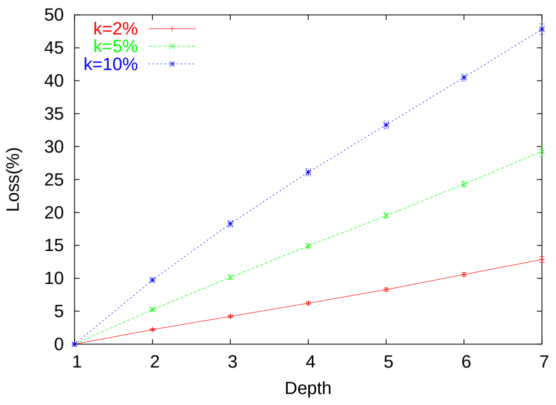
The rest of this chapter is divided into three sections. In the first section, we show the behavior of the basic non-adaptive algorithm. In the second section we verify our model by comparing observed values with those calculated from the model. Lastly, we show the performance of the adaptation algorithms in the fact of dynamic input.

6.1 Performance of Basic Algorithm

6.1.1 Effect of k on Sleep and Loss



(a) Sleep



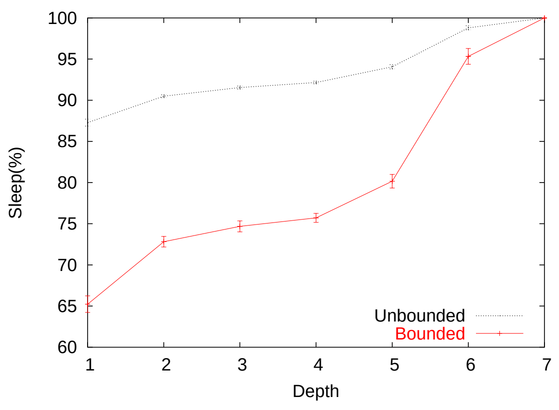
(b) Loss

Figure 6.1: Effect of k on Sleep and Loss; $\beta = 0.05$

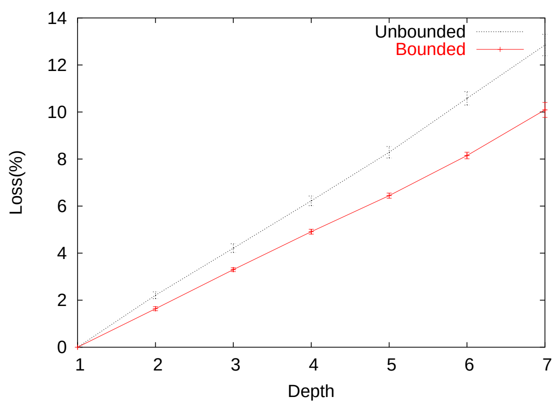
Our model predicts that beyond a point, allowing more loss will only decrease sleep; it will reduce the benefit of the algorithm. We test this by running simulations with various values of k. The non-adaptive algorithm with no change in behavior in any node (the "static" case) suffices. Figures 6.1(a), 6.1(b) shows sleep and loss respectively against node depth (number of hops from base station) for several values of k. It is seen clearly that increasing k is actually leading to less and less sleep in Figure 6.1(a).

6.2 Effect of Delay Bounds

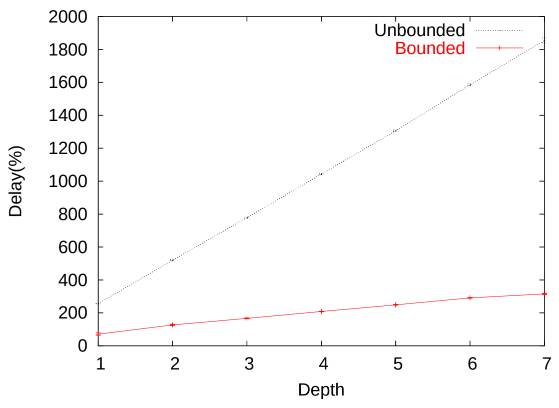
Figure 6.2 shows the results for a scenario in which every node specifies a delay bound equal to its sensing period. The delay bound forces a smaller transmission period (about 1/7 of the unbounded period in our simulations), while the spread of the inter-arrival distribution remains the same. Effectively, we have a larger value of β and hence a smaller α. 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 expected. The tradeoff is the overhead: there are significantly more dummy transmissions with delay bounds. However, the number of transmissions remains less than the case with no delay bounds in the crucial *depth one* nodes where the sleep is minimum. This means that the advantage of aggregation, which allows the inner nodes to sleep more is present in both cases.



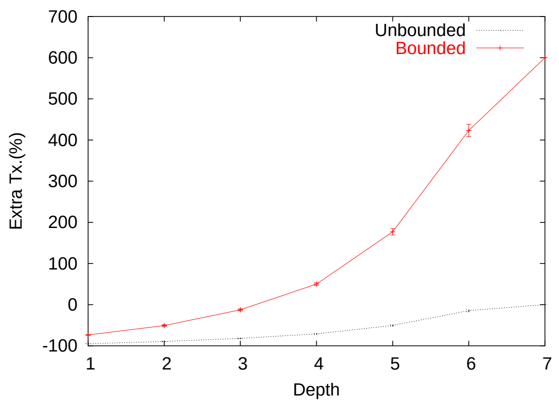
(a) Sleep



(b) Loss



(c) Delay



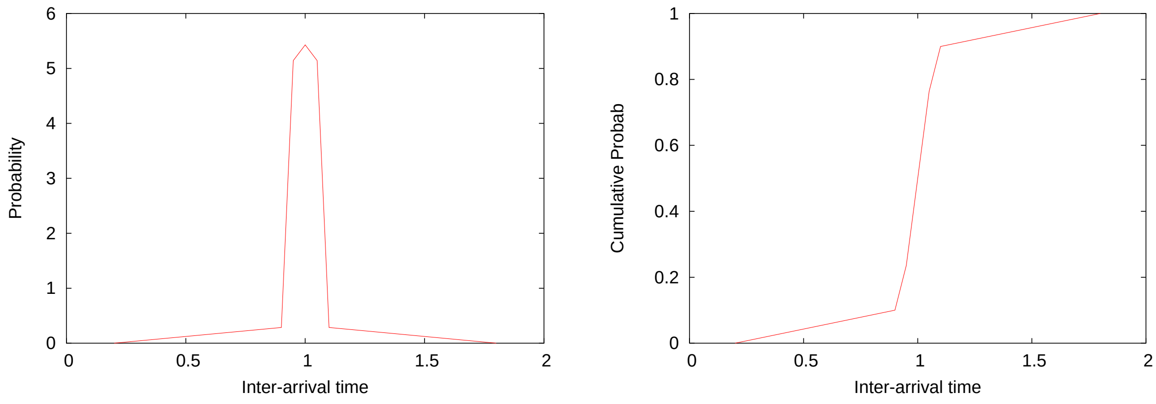
(d) Overhead

Figure 6.2: Effect of Delay Bound: comparing unbounded delays with bound equal to one period.

6.3 Model

We now verify and validate our model of loss, sleep and benefit. First we show that our intuition about an optimal value of α (or k) that maximized benefit is correct. We then compare loss and sleep values observed in simulations with values computed using our equations and show that they match well.

6.3.1 Maximum Benefit of Algorithm



(a) PDF (b) Predicted

Figure 6.3: A Triangular approximation to the normal distribution with a light but long tail

To show that benefit is maximum for some value of k, we use a triangular approximation to the normal distribution that has a long-but-light-tail as shown in Figure 6.3 to generate inter-arrival times instead of the truncated normal distribution used in other simulations. We then plot the values of loss, sleep and benefit for various values of k in Figure 6.4. We see that loss increases almost linearly, sleep increases and then decreases as expected. Benefit peaks for $k \approx 8\%$.

6.3.2 Loss Model

Figure 6.5 shown 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

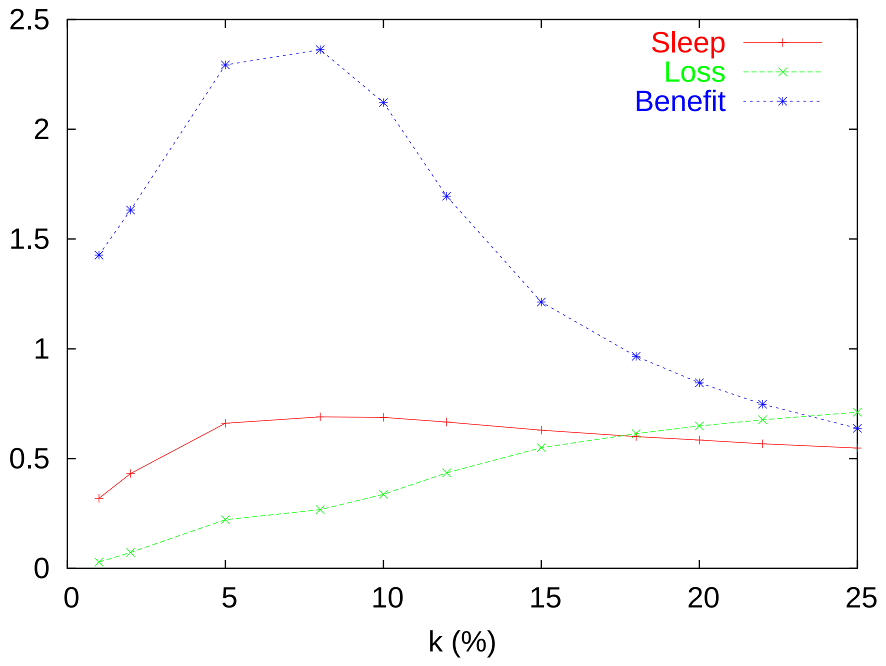
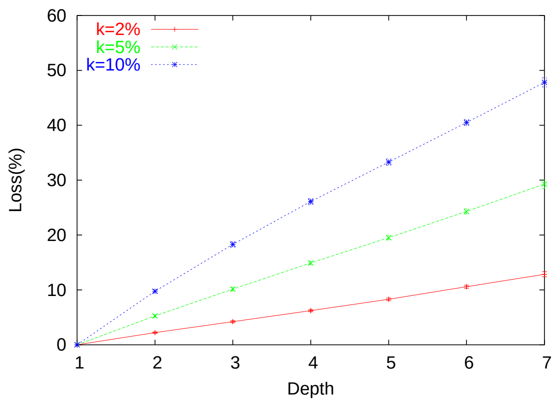
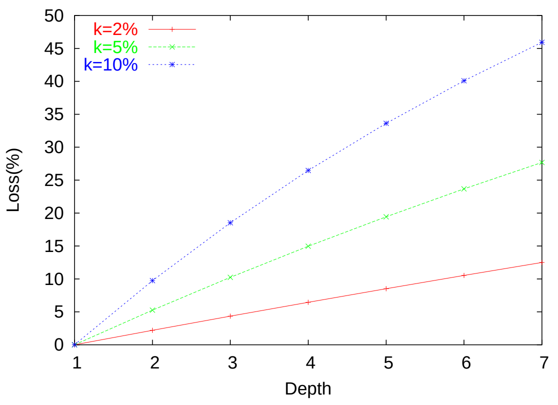


Figure 6.4: Loss, Sleep and Benefit for various values of k for the inter-arrival distribution in Figure 6.3

higher values of k, the departure from linearity is noticeable. The predicted values are calculated using the more accurate Equation 5.5 not its approximation Equation 5.6. 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.



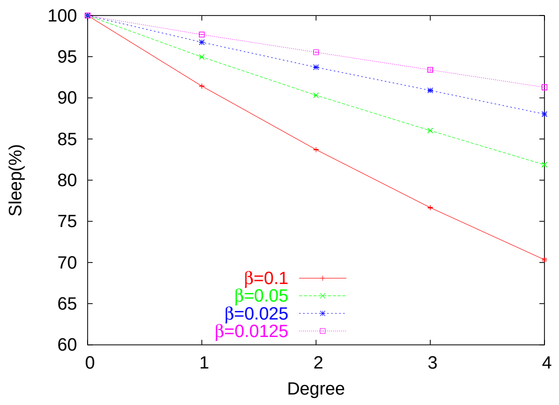
(a) Actual



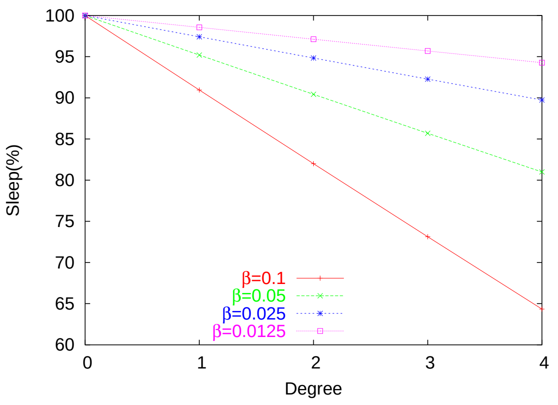
(b) Predicted

Figure 6.5: Effect of depth on Loss, for various k

6.3.3 Sleep Model



(a) Actual



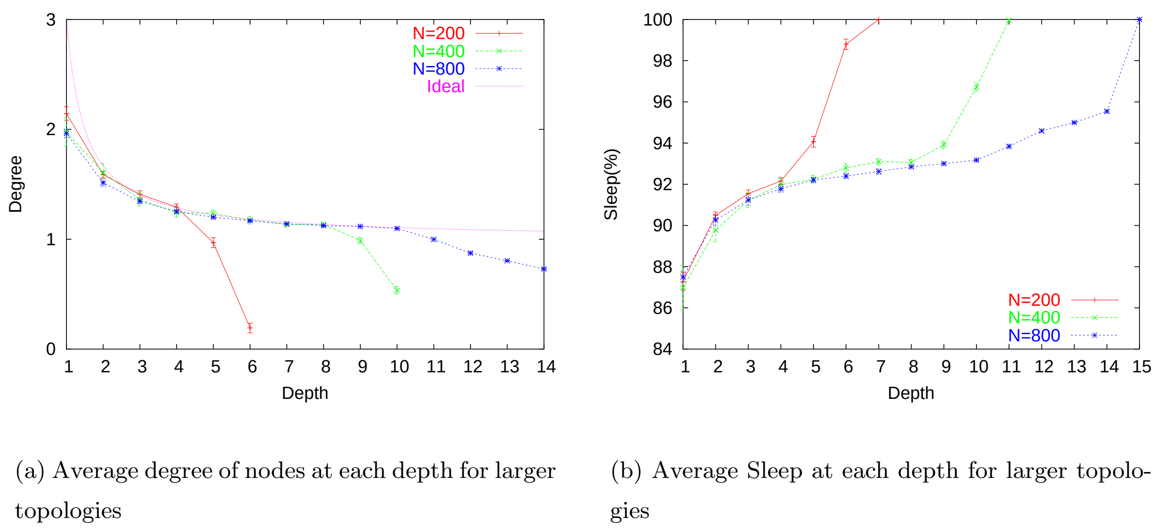
(b) Predicted

Figure 6.6: Effect of in-degree on Sleep for various β; $k = 1\%$

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 6.6(a) shows sleep against degree of a node for several values of β, 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 6.6(b). These values were calculated using Equation 5.7. The value of $F(\alpha)$ used in this equation is the average loss suffered by nodes at depth one. Using these, α 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 β and degree. This is because the approximations made in the model are not valid for large values of the degree (η) and β 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 η and β. Since this is the expected scenario in real networks, the model is useful.

Effect of Depth



(a) Average degree of nodes at each depth for larger topologies

(b) Average Sleep at each depth for larger topologies

Figure 6.7: 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. In networks with a uniform density of nodes, the average degree of nodes at a given distance from the sink can be calculated:

$$\eta_d = \frac{2d+1}{2d-1} \tag{6.1}$$

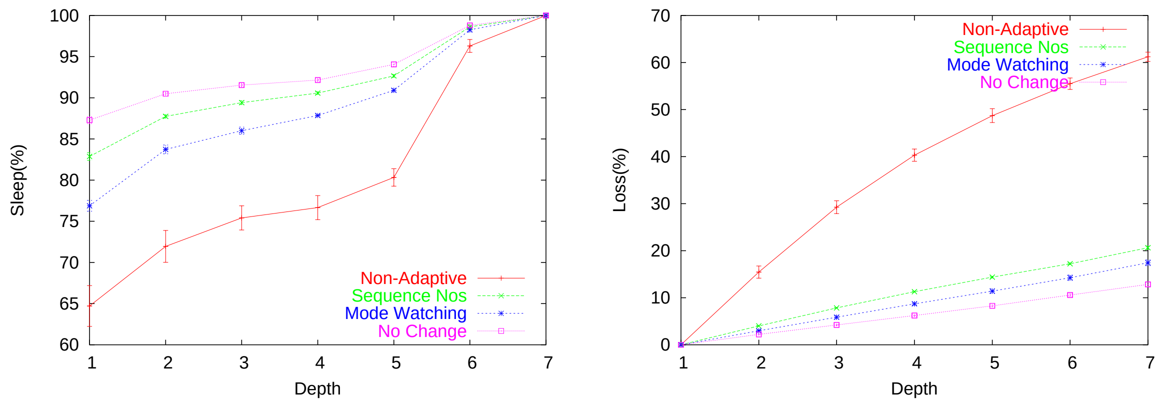
Figure 6.7(a) shows average degree at each depth for three topology sizes that we have used. It also shows the ideal value according to Equation 6.1. We see that all topologies begin in the same way and follow the ideal before dropping abruptly as the reach the fringes of the network. Figure 6.7(b) shows the average sleep attained by nodes at each depth for the same three topology sizes. Figure 6.7(b) is approximately an inversion of Figure 6.7(a) as suggested by Equation 5.7.

6.4 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 some speed up. 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.

6.4.1 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 6.8(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 6.8(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



(a) Sleep (b) Loss

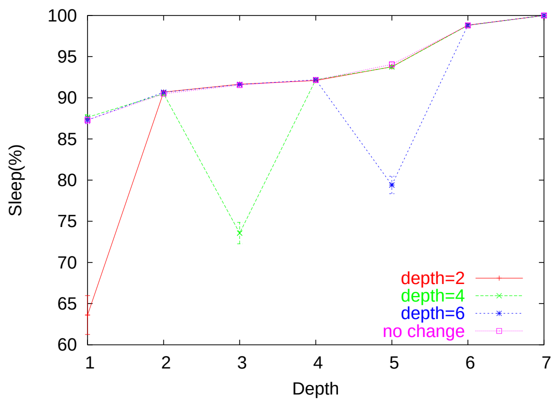
Figure 6.8: Effect of Speeding up: all algorithms

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.

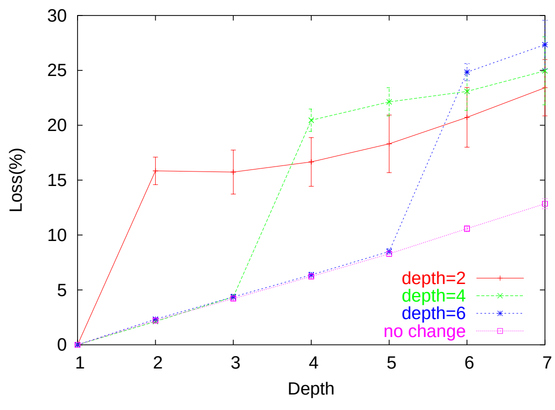
To better understand the reaction of each algorithm to speeding up, we investigate each algorithm separately, limiting the speedup to nodes at a particular depth each time.

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 of a particular depth alone. We run simulations with the depth at which the change occurs set to 2, 4 and 6 separately. Figure 6.9(a) shows the effect on sleep and Figure 6.9(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



(a) Sleep



(b) Loss

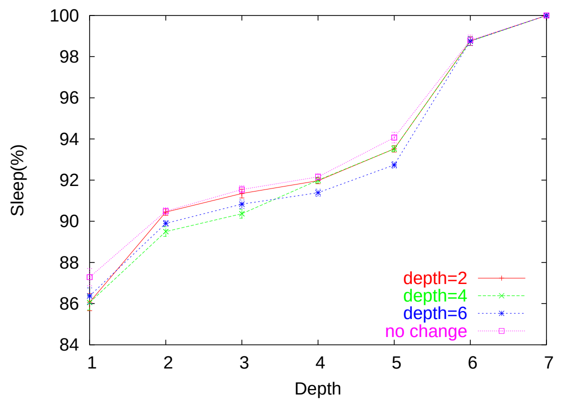
Figure 6.9: Effect of Speeding up: non-adaptive case at three depths.

nodes at the depth at which change occurs, where it may be lost.

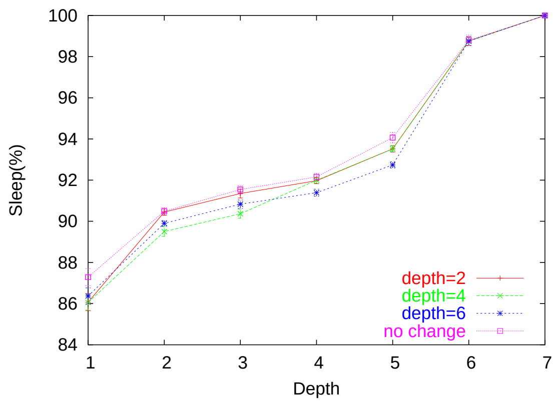
Adaptation with Sequence Numbers

Figure 6.10(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 6.10(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.



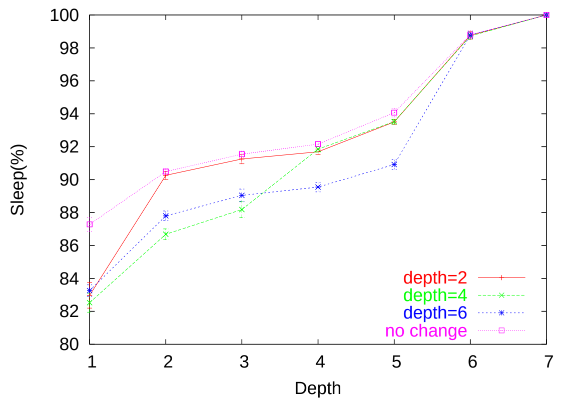
(a) Sleep



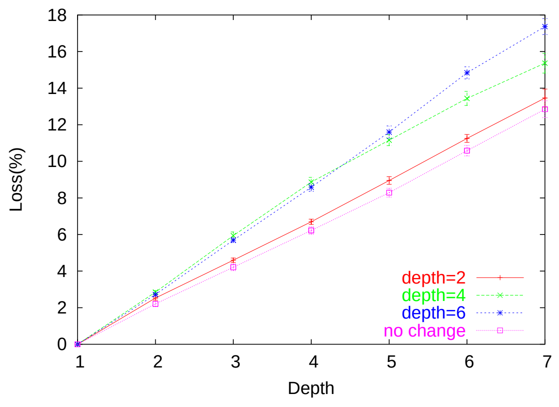
(b) Loss

Figure 6.10: Effect of Speeding up: Adaptation with Sequence Numbers, at three depths.

Adaptation by Mode Watching



(a) Sleep

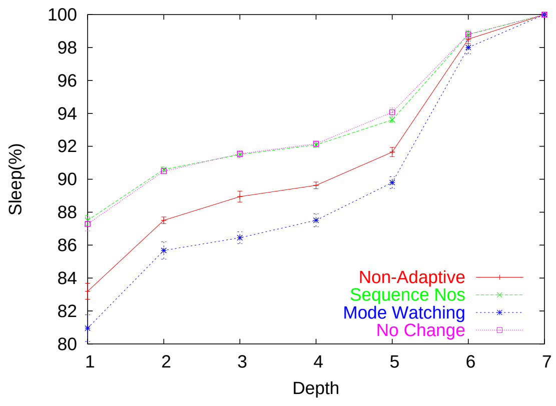


(b) Loss

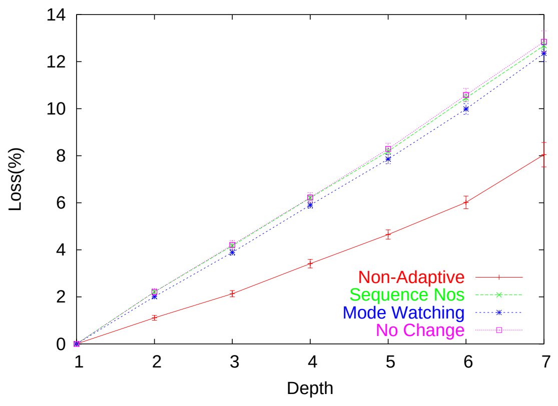
Figure 6.11: Effect of Speeding up: adaptation by mode watching, at three depths

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 6.11.

6.4.2 Effect of Slowing down



(a) Sleep



(b) Loss

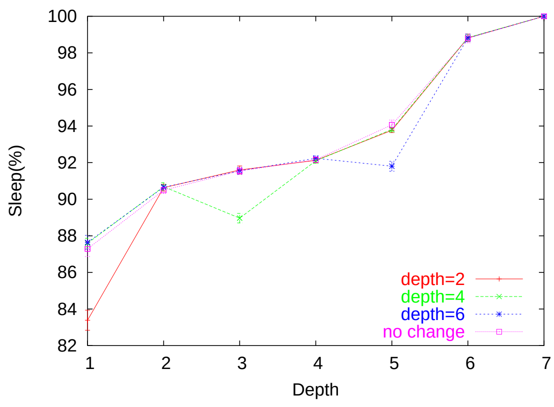
Figure 6.12: Effect of Slowing down: all algorithms

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 6.12(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 6.12(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.

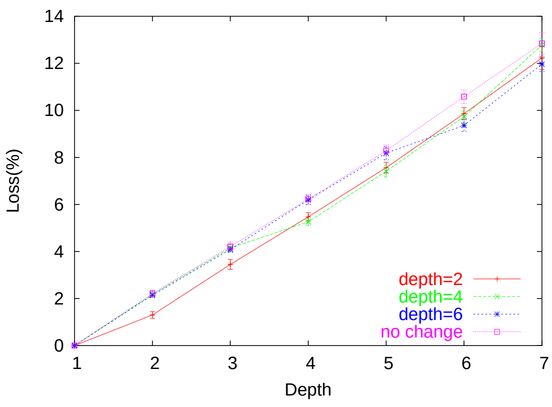
The mode observing adaptation algorithm does not perform so well. It does detect that the period has increased, however, by staying awake to learn the new behavior it is sacrificing a lot of sleep. There is no increase in loss due to this type of change and the sleep lost can significantly cut into the 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.

Non-Adaptive Algorithm



(a) Sleep

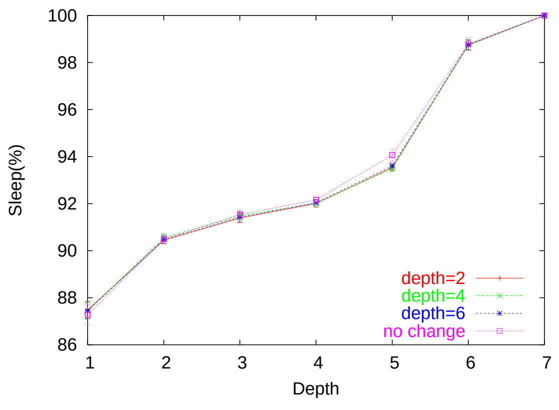


(b) Loss

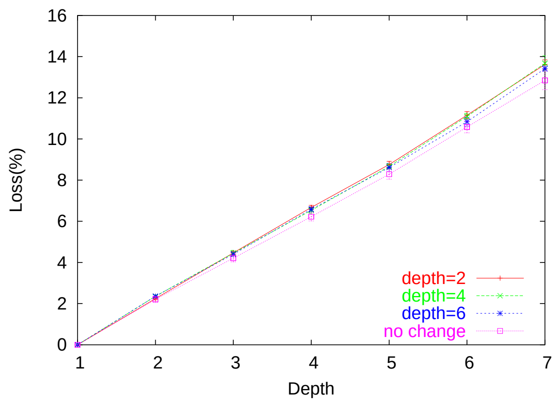
Figure 6.13: Effect of Slowing Down: non-adaptive case at three depths.

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 6.13(a) shows the effect on sleep and Figure 6.13(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.



(a) Sleep



(b) Loss

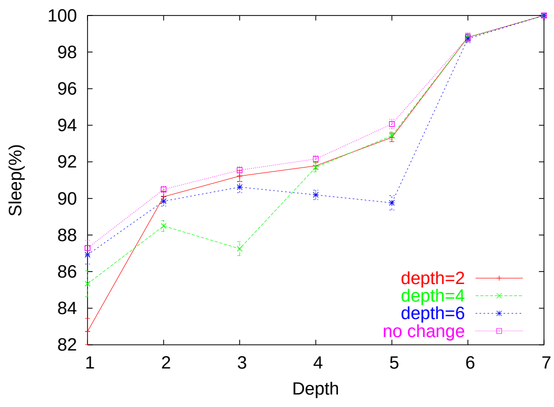
Figure 6.14: Effect of Slowing Down: Adaptation using Sequence Numbers at three depths.

Adaptation using Sequence Numbers

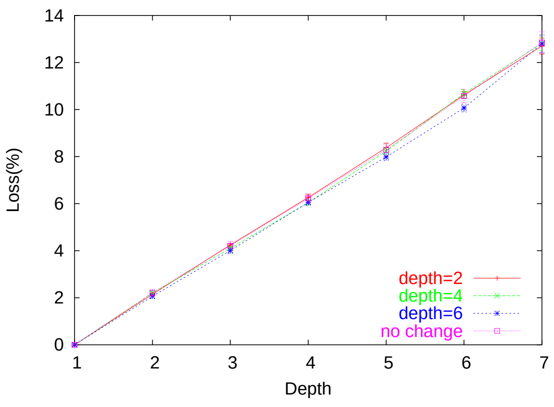
Figure 6.14 shows that adaptation by sequence numbers performs better than the non-adaptive case. It is able to detect that the inter-arrival times are longer and starts sleeping more too. Although it loses less when the change starts to happen, it loses as much as before after it has adapted to the 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 by Mode Watching

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 6.15. 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.



(a) Sleep



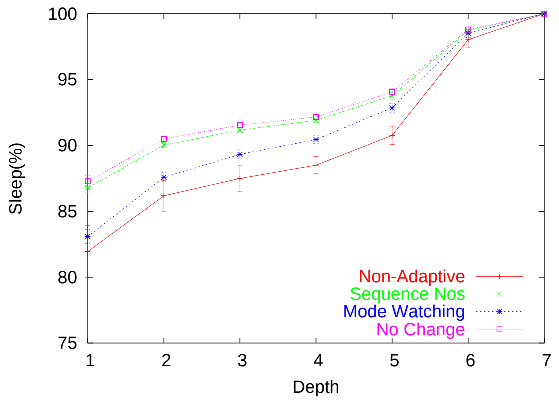
(b) Loss

Figure 6.15: Effect of Slowing Down: Adaptation by Mode Watching at three depths.

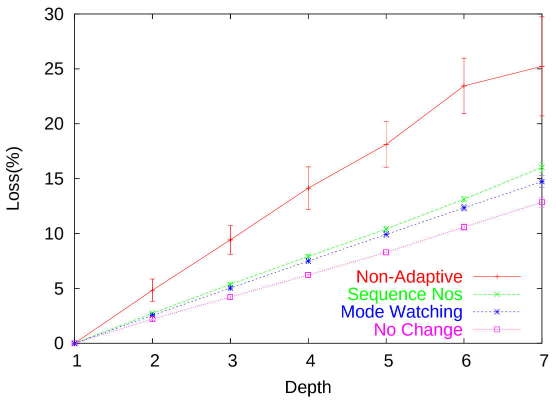
6.5 Speeding Up and Slowing Down

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 ±10% in either direction. Figure 6.16 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.

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.



(a) Sleep



(b) Loss

Figure 6.16: Effect of Change: Adaptation Algorithms Compared

Chapter 7

Conclusions

We have shown that significant power savings are possible in wireless sensor networks simply by each node observing its neighbors are learning their behavior. The parameters that affect the algorithms are identified and their effects investigated. We have also shown how nodes can adapt to changes in behavior by neighbors, which increases the applicability and utility of the algorithms in realistic situations. We have also provided a quantitative method to judge the performance of the algorithms and have shown the conditions under which the algorithms operate best.

Our model of sleep, loss and benefit also provides an insight into how the performance can or cannot be further improved. This is important because the longer the nodes closest to the sink live, the longer the network lives. We have shown, for example, that allowing a node to lose more is counter productive after a point. On the other hand, decreasing β by increasing the mean transmission period of a node's upstream neighbors will allow the node to sleep more. If the node-to-sink delay is bounded, then this bound can be non-uniformly apportioned between all the nodes on the path to the sink such that inner nodes get to sleep more.

Our study of uniform density topologies in the previous chapter has shown that beyond depth two, nodes have more or less the same in-degree: one or two. Thus depths beyond two can be treated uniformly. Only depths one and two require special attention to increase their lifetimes, which eases the task.

List of References

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *IEEE Communication Magazine*, 40(8):102–116, aug 2002.

[2] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: A survey. *IEEE Computer*, 38(4):393–422, mar 2002.

[3] Javed Aslam, Qun Li, and Daniela Rus. Three power-aware routing algorithms for sensor networks. *Wireless Communications and Mobile Computing*, 2(3):187–208, mar 2003.

[4] J.-C. Cano and P. Manzoni. Evaluating the energy-consumption reduction in a manet by dynamically switching-off network interfaces. In *Proc. of the Sixth IEEE Symposium on Computers and Communications*, pages 186–191, 2001.

[5] C.-C. Chiang, H. K. Wu, W. Liu, and M. Gerla. Routing in clustered multihop mobile wireless networks with fading channel. In *IEEE SICON '97*, pages 197–211, April 1997.

[6] I. Chlamtac, M. Conti, and J. J. N. Liu. Mobile ad hoc networks: imperatives and challenges. *Ad hoc Networks*, 1(1):13–64, July 2003.

[7] M. Scott Corson, Joseph P. Macker, and Gregory H. Cirincione. Internet-based mobile ad hoc networking. *IEEE Internet Computing*, 3(4):63–70, 1999.

[8] L[aura] M[arie] Feeney and M[artin] Nilsson. Investigating the energy consumption of a wireless network interface in an ad hoc networking environment. In *Proc. of IEEE INFOCOM*, pages 1548–1557, 2001.

[9] J.A. Freebersyser and B. Leiner. *Ad Hoc Networking*, chapter A DoD perspective on mobile ad hoc networks; Charles. E. Perkins (ed), pages 29–51. Addison Weasley, Reading, M.A., 2001.

[10] A.A. Goel. Power conservation in wireless sensor networks using switch-off. Master's thesis, North Carolina State University, 2003. Publicly available via WWW: *http://www.lib.ncsu.edu/ETD-db/*.

[11] D. B. Johnson and D. A. Maltz. Dynamic source routing protocol for ad hoc networks. In T. Imielinskli and H. Korth, editors, *Mobile Computing*, pages 153–181. Kluwer, 1996.

[12] Robin Kravets and P. Krishnan. Power management techniques for mobile communication. In *Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*, pages 157–168. ACM Press, 1998.

[13] Joseph P. Macker and M. Scott Corson. Mobile ad hoc networking and the ietf. *SIGMOBILE Mobile Computer Communication. Review*, 2(1):9–14, 1998.

[14] R[ex] Min, M[anish] Bhardwaj, N[athan] Ickes, A[lice] Wang, and A[nantha] Chandrakasan. The hardware and the network: Total-system strategies for power aware wireless microsensors. In *Proc. of the IEEE CAS Workshop on Wireless Communications and Networking*, Pasadena, CA, USA, Sep. 2002.

[15] S. Murthy and J. J. Garcia-Luna-Aceves. An efficient routing protocol for wireless networks. *ACM Mobile Networks and Applications Journal*, pages 183–197, October 1996.

[16] S[wetha] Narayanaswamy, V[ikas] Kawadia, R. S. Sreenivas, and P. R. Kumar. Power control in ad-hoc networks: Theory, architecture, algorithm and implementation of the COMPOW protocol. In *Proc. of European Wireless 2002. Next Generation Wireless Networks: Technologies, Protocols, Services and Applications*, pages 156–162, Florence, Italy, Feb. 25-28 2002.

[17] V. D. Park and M. S. Corson. A highly adaptive distributed routing algorithm for mobile wireless networks. In *INFOCOM'97*, April 1997.

[18] C. E. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computer. *Computer Communications Review*, pages 234–244, October 1994.

[19] C. E. Perkins and E. M. Royer. Ad hoc on-demand distance vector routing. In *2nd Intl. Wksp. Mobile Computing Systems and Applications*, pages 90–100, Feb. 1999.

[20] Charles E. Perkins. *Ad Hoc Networking*. Addison-Wesley, 2000.

[21] Ram Ramanathan and Regina Hain. Topology control of multihop wireless networks using transmit power adjustment. In *INFOCOM (2)*, pages 404–413, 2000.

[22] Elizabeth M. Royer and Chai-Keong Toh. A review of current routing protocols for ad hoc mobile wireless networks. *IEEE Personal Communications*, 6(2):46–55, April 1999.

[23] R. Rozovsky and P. R. Kumar. SEEDEX: A MAC protocol for ad hoc networks. In *Proc. of The ACM Symposium on Mobile Ad Hoc Networking & Computing, (MobiHoc)*, pages 67–75, Long Beach,CA, oct 2001.

[24] Eugene Shih, Seong-Hwan Cho, Nathan Ickes, Rex Min, Amit Sinha, Alice Wang, and Anantha Chandrakasan. Physical layer driven protocol and algorithm design for energy-efficient wireless sensor networks. In *Proc of Mobicom 2001*, 2001.

[25] Mihail L. Sichitiu. Cross-layer scheduling for power efficiency in wireless sensor networks. In *Proc. of Infocom 2004*, 2004.

[26] Mihail L. Sichitiu and Rudra Dutta. Benefits of multiple battery levels for the lifetime of large wireless sensor networks. In *Proc. of the Second ACM International Workshop on Wireless Sensor Networks and Applications (WSNA '03) (submitted)*, 2003.

[27] S. Singh and C. Raghavendra. PAMAS: Power aware multi-access protocol with signalling for ad hoc networks. *ACM Computer Communications Review*, 1999.

[28] S. Singh and C.S. Raghavendra. Power efficient MAC protocol for multihop radio networks. In *The Ninth IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, pages 153–157, 1998.

[29] M[ark] Stemm and R[andy] H. Katz. Measuring and reducing energy consumption of network interfaces in hand-held devices. *IEICE Transactions on Communications*, E80-B(8):1125–1131, 1997.

[30] M. W. Subbarao. Dynamic power-conscious routing for MANETS: an initial approach. In *IEEE Vehicular Technology Conference*, pages 1232–1237, 1999.

[31] A. S. Tanenbaum. *Computer Networks.* Prentice-Hall, Upper Englewood Cliffs, NJ, 4th edition, 2002.

[32] C.-K. Toh. Maximum battery life routing to support ubiquitous mobile computing in wireless ad hoc networks. *IEEE Communications Magazine*, 36(6):138–147, Jube 2001.

[33] C. K. Toh. *Ad hoc Mobile Wireless Networks and Protocols.* Prentice Hall, 2002.

[34] A. Woo and D. E. Culler. A transmission control scheme for media access in sensor networks. In *Proc. ACM MOBICOM*, jul 2001.

[35] Wei Ye, John Heidemann, and Deborah Estrin. An energy-efficient MAC protocol for wireless sensor networks. pages 1567–1576. USC/Information Sciences Institute.