

MANAGEMENT OF RESOURCE-CONSTRAINED SENSOR NETWORKS

A Thesis Presented

by

SRIKANTH SUNDARESAN

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL AND COMPUTER ENGINEERING

May 2008

Department of Electrical and Computer Engineering

© Copyright by Srikanth Sundaresan 2008

All Rights Reserved

MANAGEMENT OF RESOURCE-CONSTRAINED
SENSOR NETWORKS

A Thesis Presented

by

SRIKANTH SUNDARESAN

Approved as to style and content by:

C. Mani Krishna, Chair

Israel Koren, Member

Aura Ganz, Member

C.V. Hollot, Department Chair
Electrical and Computer Engineering

ACKNOWLEDGMENTS

I would like to thank my advisor Professor C. Mani Krishna for his patience and encouragement at all times and of course for his guidance. I would also like to thank Professor Israel Koren and Dr. Zahava Koren for their valuable inputs. Special thanks also to my lab-mates for helping me with key concepts and also providing light relief in equal measure.

ABSTRACT

MANAGEMENT OF RESOURCE-CONSTRAINED SENSOR NETWORKS

MAY 2008

SRIKANTH SUNDARESAN

B.Tech., MALAVIYA NATIONAL INSTITUTE OF TECHNOLOGY, JAIPUR
INDIA

M.S.E.C.E., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor C. Mani Krishna

Energy conservation is one of the most important issues in sensor networks. We consider the problem of maximizing performance benefit of energy constrained sensor networks in the context of event detection applications. We propose two adaptive event-driven duty cycling techniques that change the node's duty cycle based on event arrival patterns. We show that such an approach is very useful both in cutting down wasteful energy consumption as well as improving the detection performance in many practical applications where there are long periods of lull followed by a burst of activity. We demonstrate the techniques' effectiveness over a static duty cycling technique as well as another adaptive scheme PEAS [22] using as our performance metric a function of the number of missed events and false alarms . The techniques proposed are simple to implement at the node level and at the same time provide significant benefits in both energy efficiency as well as performance.

TABLE OF CONTENTS

	Page
ACKNOWLEDGMENTS	iv
ABSTRACT	v
LIST OF TABLES	viii
LIST OF FIGURES	ix
CHAPTER	
INTRODUCTION	1
1. RELATED RESEARCH	4
2. ADAPTIVE DUTY CYCLING	6
2.1 Simple Adaptive Duty Cycling	6
2.2 Adaptive Algorithm based on Markov Decision Theory	7
3. THE SYSTEM	10
3.1 Nodes	10
3.2 The Environment Model	11
3.3 The Event Model	12
3.4 Estimating Cost	12
4. RESULTS AND ANALYSIS	13
4.1 Simulation Settings	13
4.2 Results	14
4.2.1 Effect of Starting Energy	14
4.2.2 Effect of Node Density	16
4.2.3 Transmission Radius and Detection Threshold	18
4.2.4 Effect of Event Clustering	20

4.2.5	Effect of Mission Time	22
4.2.6	Duration of Environment	23
4.2.7	Node Distribution	24
4.2.8	Application-specific requirements	25
4.2.9	Benefit of Adaptivity	26
4.2.10	Effect of MDT Parameters	28
4.2.11	Effect of Rate of Adaptation	30
4.3	Fault Tolerance	32
5.	CONCLUSION AND FUTURE WORK	34
	APPENDIX: THE SIMULATOR	36
A.1	About the Simulator	36
A.2	Installation and Compilation	37
A.3	Examples	38
	BIBLIOGRAPHY	41

LIST OF TABLES

Table	Page
4.1 Parameter values used	14
4.2 Effect of Table Compression	32
A.1 Parameter Names	39
A.2 Data Structures	40
A.3 Important functions	40

LIST OF FIGURES

Figure	Page
3.1 Environment model	11
4.1 Effect of starting energy on cost	15
4.2 Effect of starting energy on node lifetime	15
4.3 Effect of starting energy on missed events	16
4.4 Effect of node density on cost	17
4.5 Effect of node density on missed events	17
4.6 Effect of node density on node lifetime	18
4.7 Effect of transmission radius on cost	20
4.8 Effect of detection threshold on cost	21
4.9 Effect of event clustering	22
4.10 Effect of mission time on cost	23
4.11 Effect of mission time on node lifetime	24
4.12 Effect of environment duration	24
4.13 Effect of node distribution density	25
4.14 Effect of Application-specific requirements	26
4.15 Effect of starting wake fraction on cost	27
4.16 Effect of starting wake fraction on node lifetime	28
4.17 Effect of MDT Parameters	30

4.18	Effect of minimum wake fraction on cost	31
4.19	Effect of minimum wake fraction on adaptivity	31
4.20	Effect of rate of adaptation on cost	32
4.21	Effect of rate of adaptation on node lifetime	33

INTRODUCTION

Sensor networks are used in a wide range of applications, from observing the environment [20], to studying animal behavior [10], to detecting intrusions [1]. Apart from sensing capabilities, each node also has limited processing and communication capabilities. Due to the small size and low cost of the components, nodes can be deployed on a large scale and in inhospitable terrain at no great cost. A major challenge in managing such systems is their energy-constrained nature: they are generally battery-powered with no means to recharge or replace the batteries once depleted. Managing the limited energy resource is of primary importance in ensuring the usefulness of such systems.

A common method for conserving energy and extending the system lifetime is *duty cycling*. Nodes stay awake only a fraction of the time; sensing and communication can only be done when the node is awake. Duty cycling achieves the object of prolonging lifetime, and has been studied extensively in the context of monitoring and periodic sampling applications. A lot of work has gone into making the communication between nodes effective and efficient [2, 4, 5, 19]. However, event detection applications such as for intrusion, lightning strikes and fires present a different set of challenges [12], especially when these events occur rarely. In many applications, the overall frequency of events is low, but they tend to occur in clusters. In such cases the node has to try and adapt its duty cycle so that it sleeps more during periods of inactivity and less during periods of activity. Many approaches have dealt with the rare event problem [7, 12]. They deal with it in novel ways, by using radio waves to awaken nodes [7] or by using extremely low-power modules to awaken more

power hungry modules during the occurrence of an event. Most of them rely for their efficiency on specialized hardware and/or specific characteristics of the application.

Our contributions in this work are two novel, generic event-driven adaptive duty cycling techniques for use in rare event detection applications. No extra hardware is required. Our simple adaptive technique learns and adapts its duty cycle based on the arrival patterns of events. We also propose a more sophisticated heuristic based on Markov Decision Theory that controls the wake/sleep pattern of nodes based on event arrival patterns. The techniques offer significant advantages in performance, energy efficiency and also generality; using the arrival pattern of events, a sensor network system can optimize its performance as well as its lifetime. We measure performance as the total expected cost of the system in terms of missed events and false alarms. Every missed event and false alarm has a cost associated with it and we aim to minimize the total cost of the system during its designated period of operation. The cost is defined according to the requirements of the application. The expected cost is affected by various system parameters especially:

- *Energy*, which determines how constrained the system is and how important energy conservation algorithms are.
- *Duty cycling*, which provides the fine balance between sensing performance and system longevity.
- *Node density*, which might be used to cut down on both missed events and false alarms by requiring multiple, corroborating reports of events.
- *Event characteristics*, which determine how useful an event-driven adaptive scheme can be; if a pattern exists which the scheme can adapt to, the benefits can be quite substantial.

We study these parameters and how they interact with each other to affect the cost of the system. We show that the adaptive schemes provide significant gains, *with*

minimal tuning, over non-adaptive schemes whose best performance occurs over a narrow range of parameter settings. We also compare our heuristics with the Probing Environment and Adaptive Sleeping (PEAS) algorithm [22] and show that they outperform it when event clustering occurs. PEAS probes the neighborhood to see if nodes are awake, and wakes up only when there are no nodes awake in the neighborhood. We use PEAS because it is one of the few adaptive algorithms available in literature. The main purpose behind PEAS is robustness and reliability. We describe the PEAS algorithm in more detail in Chapter 1.

The rest of the thesis is organized as follows: Chapter 1 deals with related research, Chapter 2 describes the proposed heuristics, Chapter 3 explains the model used for the simulation, Chapter 4 lists simulation settings and presents results, and finally Chapter 5 concludes the thesis.

CHAPTER 1

RELATED RESEARCH

A lot of research has been carried out in the field of sensor networks; particularly performance and energy efficiency. Performance has been studied from the point of view of energy [3, 16, 17, 23], network issues [3, 5, 19, 24], and quality of sensing [9] among others. Zhao *et al* [24] study network performance in terms of packet delivery in different environments. The issue of energy efficiency has been addressed at various levels in the sensor network architecture. Sinha *et al* [17] deal with a system with multiple sleep states and employ a transition policy which uses event prediction to decrease energy consumption. They also propose Dynamic Voltage Scaling (DVS) to reduce energy consumption. Energy scaling algorithms have been studied in [16] where the trade-off between processing quality and energy efficiency offered by these algorithms is examined. Various MAC protocols have been introduced [2, 14, 19, 23], which try to reduce the power consumed due to medium access. These techniques are useful in applications where communication is frequent and dominates the energy consumption. Some authors have also studied applications where communication is rare. Ganeriwal *et al* [5] propose efficient radio communication techniques for such cases. In general, for applications that sense and report events that only occur rarely and aperiodically, we also need to concentrate on conserving energy in other ways. Adaptive duty cycling has also been studied in one form or the other, in [7, 12] amongst others. Dutta *et al* [12], and Gu *et al* [7] propose architectures that provide wake mechanisms for nodes when events occur. In the former, low energy sensors stay awake throughout and wake the more energy hungry units as required. In the

latter, the authors propose a radio wake up circuit that uses the electromagnetic energy from radio waves to wake up the node. There is no periodicity involved in either, and both achieve the goal of reducing the idle wake period when there are no events occurring, however they are both application specific. [6, 22] propose energy aware adaptive schemes. In both, nodes wake up only when they do not sense nodes awake in the neighborhood. In the former, nodes probe its immediate neighborhood to see if any node is awake, If it sees that a node is awake, it goes back to sleep for an exponentially distributed amount of time. Else it stays awake till it dies. In the latter, nodes wake up only if its geographical range is not covered by other nodes. Both are essentially same in that they keep redundant nodes asleep, however, they still waste energy by keeping nodes awake when they don't really have to, for example when the probability of an event occurring in the near future is very low. We compare our schemes with [22] and show that a much improved performance can be attained when event clustering occurs by using knowledge of external environmental states. To the best of our knowledge, there is no other scheme that adapts itself to event arrival patterns. The main purpose of PEAS is reliability and robustness, while our adaptive schemes take a calculated risk of missing a few events in order to prolong lifetime and improving overall performance.

CHAPTER 2

ADAPTIVE DUTY CYCLING

2.1 Simple Adaptive Duty Cycling

In many event detection applications, there are long periods of inactivity followed by a burst of activity. If nodes can distinguish between periods of activity and inactivity, they can adapt their duty cycle accordingly. The adaptation process may vary in complexity; the technique we present bases its decision on the evidence of its current wake cycle. The simplest form of the algorithm is shown below:

```
/*initial wake_fraction in [f,1], f is the smallest fraction allowed*/
if (n > 0 events observed in current wake cycle)
    wake_fraction += n $\delta_{inc}$ 
    if (wake_fraction > 1) wake_fraction = 1
else
    wake_fraction -=  $\delta_{dec}$ 
    if (wake_fraction < f ) wake_fraction = f
```

In its simplest form, if a node sees n events in its current wake cycle, it increases its wake fraction by $n\delta_{inc}$, believing itself to be in a high activity period and hoping, as a result, to see more events in the immediate future. If it sees no events in its current cycle, it reduces its wake fraction for the next cycle by δ_{dec} , to conserve energy. Such adaptation is made subject to the requirement that the wake fraction can never drop to zero (if it did, the node would in effect be taking itself permanently out of the

network since it would have no opportunity to increase its wake fraction as a result of observing the environment). The wake fraction is trivially upper-bounded by 1.

The trade-off here is obviously between staying awake to see more events and going to sleep to conserve energy. The decision that a node makes when it adjusts its duty cycle is on the state of the external *environment*. The environment can be defined as the triggering force for the events. As mentioned earlier, in many applications, there are long periods of inactivity followed a burst. We define the environment during periods of higher activity as being *hostile* and during periods of low activity as being *benign*. In effect, the detection of an event makes the node believe that the environment is hostile. This decision is generally accurate when events arrive in clusters and when the rate of arrival in the hostile environment is much greater than the rate of arrival in the benign state. Even if the decision is false and the event was a one-off, the node does not lose much because it will get back to a lower wake fraction in the next cycle. Rare event arrival rates follow a Poisson distribution, so the benefit of event-driven adaptivity manifests itself when there is a reasonable difference in the event arrival rates in the environmental states. If there is little difference, a node in the adaptive scheme will have, at the time of dying, spent the same amount of time in the wake state as one in the static scheme, the starting energy in both being equal, although they might die at different times. Since the inter-arrival time between events is exponentially distributed, and arrivals are independent of each other, no benefit is accrued from adaptivity.

2.2 Adaptive Algorithm based on Markov Decision Theory

A *reward* is associated with every event that a node sees. A real event is given a positive reward and a false alarm is given a negative reward. Thus, the accumulated reward during a time-slice is simply the difference between the weighted probabilities of a real event and a false alarm. The weights determine the relative cost of a missed

event and a false alarm. We propose a heuristic using Markov Decision Theory in order to maximize the expected reward over the lifetime of the system. Each node decides at the beginning of each time-slice whether to wake up for the rest of the cycle or not. Calculating the appropriate action that a node should take is computationally intensive, hence we assume that the calculations are done offline in a central base station and that each node is provided with a decision table which it uses in order to look up the action it should take. The appropriate actions are picked based on current energy levels and also the current perception of the external environment. Individual nodes perceive the external environment as the probability that the external environment is hostile. Each node starts off with a (low) initial probability. If it sees an event during the current cycle, it updates the probability to the next higher value in a set. If it doesn't see an event, it chooses the next lower value.

The node chooses its action from a set of actions, based on its current state and the mission time remaining. The action is the wake probability of the node for the next cycle based on the perceived value of taking that action. We define the *state* to be a function of the number of energy units left in a node and the perception of the external environment; a unit of energy is the smallest quantum used in the node. Possible quanta that can be defined could be the energy required to stay awake for 1 time unit, or the amount required for receiving/transmitting a message. The size of the table depends on the number of states, mission time and the granularity of the environment perception values. We can compress the table based on time or state aggregation depending on the amount of memory available to the node.

Nodes report events to the base station which then decides whether the reported event is real or a false alarm. This decision is based on the number of nodes reporting an event; if it exceeds a specific threshold, then the base station declares an event. Otherwise it ignores the report. This heuristic tries to ensure that just enough nodes

stay awake to detect and report an event such that the base station declares the event. By randomizing the wake of each node, we prevent nodes in a locality waking up and sleeping in a synchronized manner, thereby creating blind spots in the system. The reason why this could happen is that it is highly likely that neighboring nodes will see the same event arrival pattern. The equations used to pick the appropriate actions are:

$$\begin{aligned}
 V(s, t) &= \max_a \left(R(s, a) + \sum_{s'} P_a(s \rightarrow s') V(s', t - 1) \right) \\
 R(s, a) &= C_{ME} P(\text{event is seen by at least } \Theta \text{ nodes} | \text{event occurs in cycle}) \\
 &\quad - C_{FA} P(\text{noise is seen by at least } \Theta \text{ nodes} | \text{noise occurs in cycle})
 \end{aligned}$$

a is the action taken at state s at a given time from a set A ,

s is the current state, a function of residual energy and perception of external environment,

s' is the state resultant from taking action a ,

t is the time remaining in the mission,

$V(s, t)$ is the expected value at state s with t time remaining,

$R(s, a)$ is the expected reward by taking action a ,

Θ is the threshold of nodes required for the base station to declare an event,

C_{ME} is the cost of a missed event,

C_{FA} is the cost of a false alarm.

CHAPTER 3

THE SYSTEM

This chapter explains the sensor network and environment model used for the simulations.

3.1 Nodes

Nodes are uniformly and randomly distributed over a given area. They have un-coordinated and adaptive wake/sleep cycles; the start of their wake cycles are independent of one other. A node detects an event if it sees a signal whose amplitude is greater than a specified threshold, θ . If an event is detected, the node sends a message announcing this to the base station and. Any appropriate routing algorithm can be used to implement the routing. Nodes are stationary. They have a finite, ideal battery. An ideal battery's performance does not degrade with residual energy level. Nodes consume energy while staying awake, and also during communication. We ignore the energy overhead in incorporating duty cycles. Such an assumption is valid because of the simplicity of the heuristics and also due to the fact that duty cycling is conceptually simple; only a timer need be set.

The base station collects reports from various nodes and decides whether an event has occurred or not. The decision is made by checking whether the number of reports it receives for an event exceeds a specified threshold.

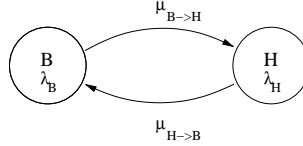


Figure 3.1. Environment model

3.2 The Environment Model

The environment is modeled as consisting of two states, *hostile* and *benign*. The states are modeled by a Markov process as shown in Figure 3.1. The system stays in either state for an exponential amount of time. All events follow a Poisson process, but their rate of arrival depends on the state of the environment. The rate of event arrival in the benign state is λ_B , while the rate in the hostile state is λ_H . Events are more likely to occur during the hostile state than during the benign state, hence typically $\lambda_H > \lambda_B$. The environment changes from benign to hostile with rate $\mu_{B \rightarrow H}$ and from hostile to benign with rate $\mu_{H \rightarrow B}$. Typically, the environment is in the hostile state only for a small fraction of the time, and the bulk of the events occur in that state. This simple model is good for many practical applications. For example, in a system detecting lightning strikes, the environment is hostile only during a thunderstorm. Lightning usually occurs during a thunderstorm, but the environment is in that state for only a small fraction of the time. However, the environment need not be restricted to exponential sojourn times; it can easily be extended so that these two states can themselves consist of multiple states in order to approximate various non-exponential sojourn time states. The duration of the environment is usually long enough for adaptivity to take place as physical events are being sensed while the time slices used for duty cycling can be of much smaller time scales.

3.3 The Event Model

Events of interest follow a given distribution for signal characteristics (amplitude and attenuation). Noise events are of two kinds, *internal* and *external*. External noise arises from some external source and can potentially be seen by multiple nodes. It arrives at a rate of λ_{ext} . Internal noise is generated within the node circuitry itself and can be seen only by the node in which it occurs. It has an arrival rate of λ_{int} . Noise is modeled as independent of real events. The signal characteristics of noise and real events may follow separate distributions. External and internal noise events are also modeled as independent of each other.

Events can occur randomly anywhere in the field, according to some distribution. The signal amplitude attenuates according to a power α of distance. The field is assumed to be even, without any physical obstructions. A node has no way of distinguishing real events from noise. It simply reports any event whose amplitude is detected to be greater than θ .

3.4 Estimating Cost

Missed events and false alarms have costs associated with them, which are application-dependent. The total cost is simply the weighted sum of the number of missed events and the number of false alarms:

$$\text{Total Cost} = C_{ME} \times N_{ME} + C_{FA} \times N_{FA} \quad (3.1)$$

where C_{ME} and C_{FA} are the costs per missed event and false alarm respectively, N_{ME} and N_{FA} are the numbers of missed events and false alarms respectively.

CHAPTER 4

RESULTS AND ANALYSIS

4.1 Simulation Settings

We assume that nodes know their own position and also that of their immediate neighbors and the base station. GPS or other means [11, 15, 13, 18] can be used to achieve localization. The routing protocol used is *Greedy Perimeter Stateless Routing* [8], which greedily passes on the message to the neighbor which is geographically closest to the base station. It is used because it is simple and guarantees delivery if the network is connected. If the node happens to be at a local minimum, that is, it has no neighbor which is geographically closer to the base station than itself, then it routes the message around the perimeter of the *face* of the polygon using the well known *right-hand rule*, till it reaches either the base station or a node which is geographically closer to the base than the node which originated the perimeter routing. If the hop count of the message exceeds a given threshold, the message is dropped. To save energy, nodes may aggregate reports of the same event from different nodes and forward a single, aggregated message. The communication channel between the nodes and the base is assumed to be error-free. The radio may or may not be duty cycled; it is independent of the sensor duty cycle that is the focus of this thesis. Efficient radio communication between nodes with low duty cycles has been studied and implemented elsewhere [5]. Just before a node runs out of energy and dies, it communicates the fact to its neighbors, so that its neighbors can adjust their routing metric accordingly. Nodes die only when they run out of energy. We consider the simplest 2-state environmental model as shown in Figure 3.1. Real and

noise events' signal amplitude follow a normal distribution with different parameters (Table 4.1) and attenuate with the square of distance. Noise events, of either kind, are not affected by the environment.

The values of the various parameters used in the simulations (unless otherwise stated) are those given in Table 4.1.

Parameter	Value
Arrival rate of events in hostile state, λ_H	4
Arrival rate of events in benign state, λ_B	0.1111
Arrival rate of external noise, λ_{ext}	0.05
Arrival rate of internal noise, λ_{int} (per node)	0.001
Transition rate from hostile to benign state, $\mu_{H \rightarrow B}$	0.2
Transition rate from benign to hostile state, $\mu_{B \rightarrow H}$	0.01056
Communication energy	1
Base Station Threshold for declaring an event	2
Awake energy (per unit time)	0.1
Incrementing step-size for adaptive algorithm, δ_{inc}	0.15
Decrementing step-size for adaptive algorithm, δ_{dec}	0.15
Avg. area in which noise can be seen by node	20.9%
Avg. area in which events can be seen by node	62.8%
Hostile environment probability set used by MDT	{0.1,0.5,0.95}
Action set used by MDT	{0.1,0.2,0.3,0.7,0.8,0.9}

Table 4.1. Parameter values used

4.2 Results

4.2.1 Effect of Starting Energy

As expected, the benefit of adaptivity is most prominent when the system is constrained for energy as seen in Figures 4.1, 4.2 and 4.3. The simple adaptive scheme conserves energy by reducing its wake fraction when it believes that the outside environment is benign, while the MDT scheme tries to make an informed decision based on event characteristics and also previous history.

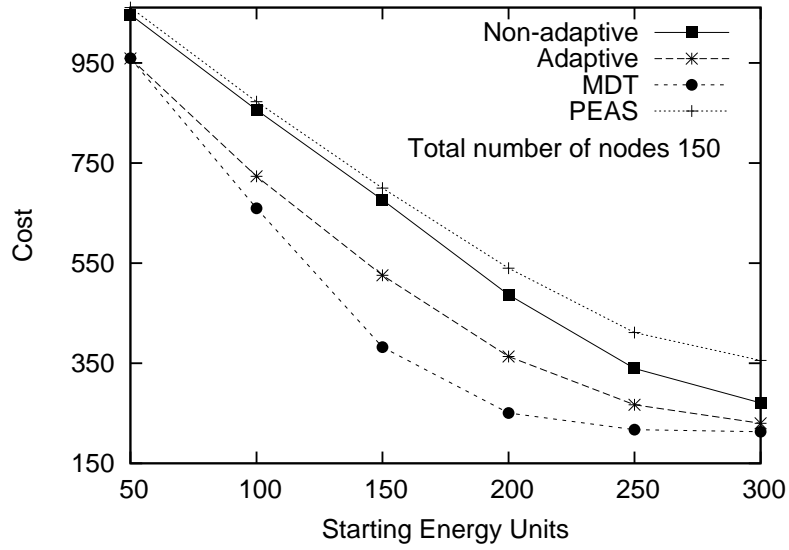


Figure 4.1. Effect of starting energy on cost

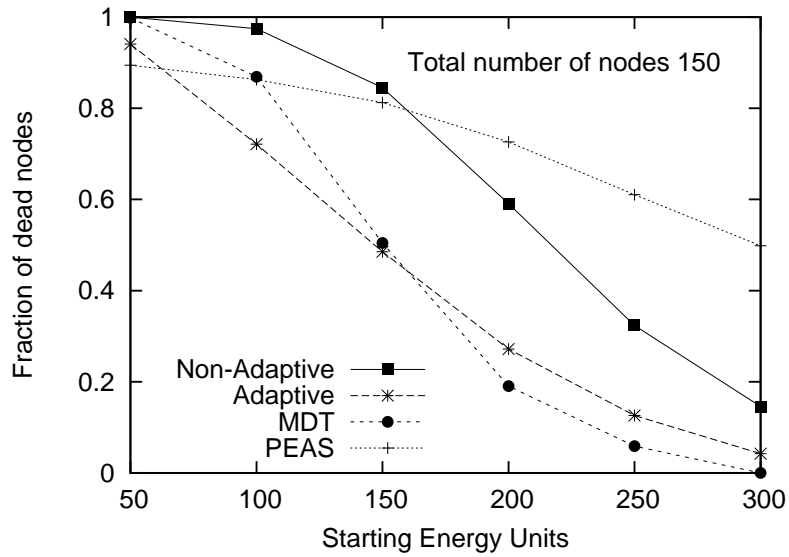


Figure 4.2. Effect of starting energy on node lifetime

The fraction of dead nodes in either scheme is a good indicator that the adaptive schemes manage to conserve energy more effectively than the static scheme or PEAS. The performance benefit of the adaptive schemes are best seen in the middle portion of the graph, where nodes don't start off with either too little or too much energy. The performance of the adaptive schemes do not improve at higher energy levels,

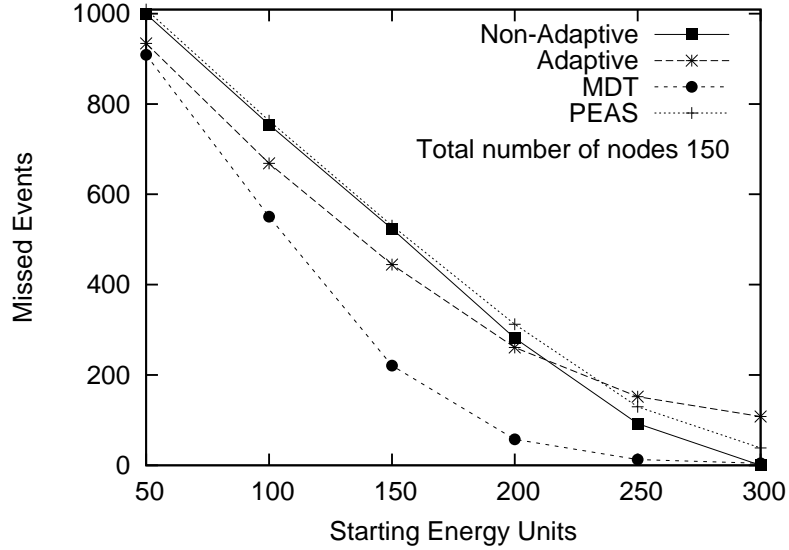


Figure 4.3. Effect of starting energy on missed events

because when energy is plentiful for the given mission time, there is little benefit in conserving energy. However the region of the graph with abundant energy is not very interesting as sensor networks do not usually operate in that region. We see that PEAS performs worse than the adaptive schemes, this is because it doesn't look to conserve energy by adapting itself according to event arrival patterns. Once nodes wake up, they stay awake till they die which is a waste of energy as it is not very productive for the bulk of the time. The simple adaptive scheme's benefit reduces somewhat when we look at only missed events, because it is by nature conservative with energy.

4.2.2 Effect of Node Density

Increased node density translates to an increased probability that some node will see an event. Hence the performance of all schemes initially improve with increasing node density. However, false alarms also increase for the same reason. Hence overall cost starts increasing after a point. This increase is lowest in the simple adaptive scheme because it is very effective in keeping out false alarms; the price being more

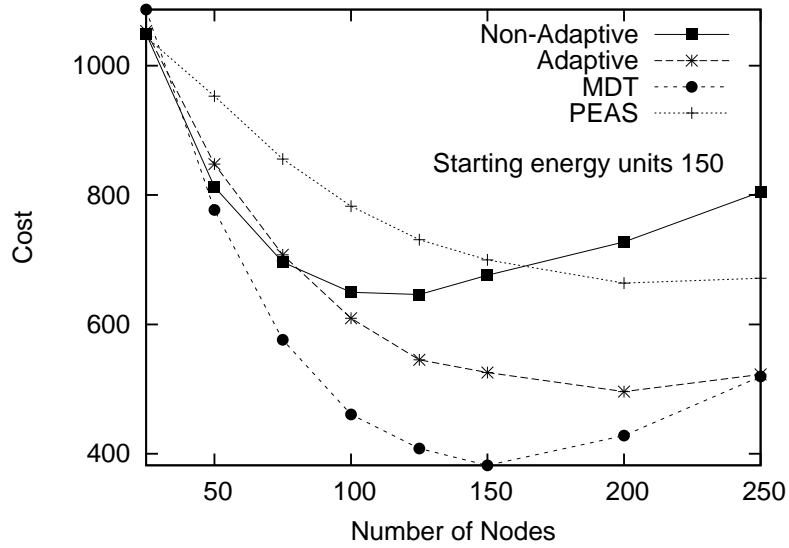


Figure 4.4. Effect of node density on cost

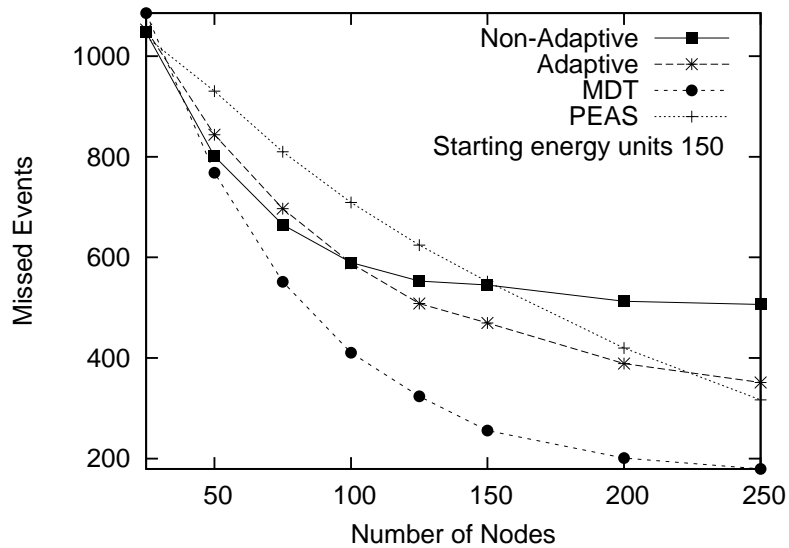


Figure 4.5. Effect of node density on missed events

missed events compared to the MDT scheme. To see the effect of MDT parameters (especially that of minimum wake probability on false alarms) see Section 4.2.10. The number of missed events decreases steadily for all schemes. Since there is localized learning in the case of the adaptive schemes, the density of nodes required by the adaptive schemes to match (or better) the performance of the static scheme is sig-

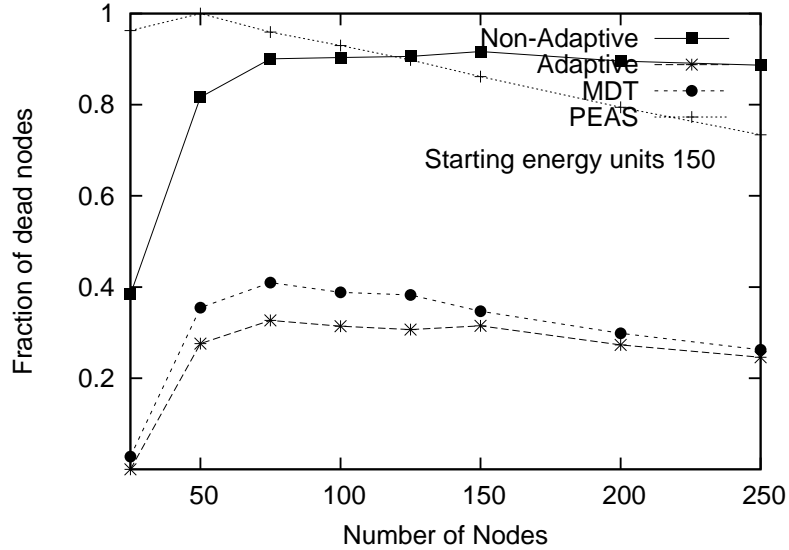


Figure 4.6. Effect of node density on node lifetime

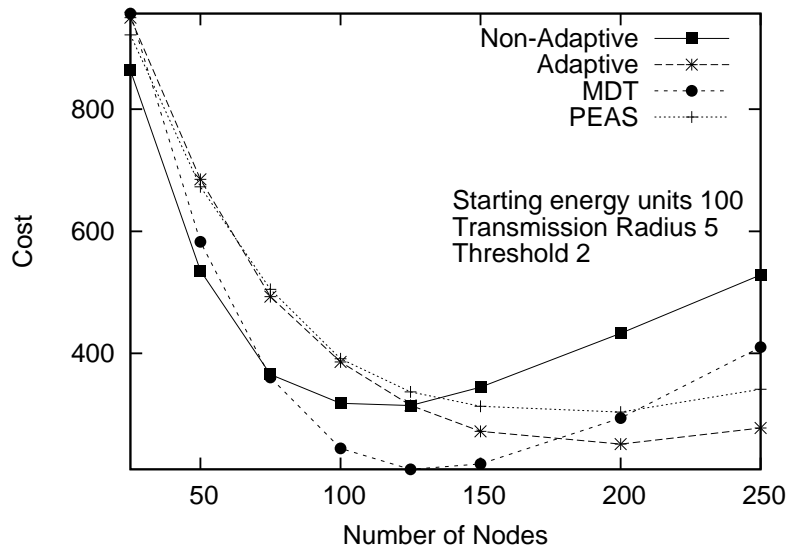
nificantly lower, as shown in Figures 4.4, 4.5, and 4.6. We see that at initially, the fraction of dead nodes increases with increasing node densities. This is because, at such low densities, even if nodes might see an event, they will not be able to communicate to base station. Since there is little communication (which forms the bulk of energy consumption) taking place, not many nodes die out. We see that performance cost is very poor at such densities. For the simple adaptive schemes, this is due to the wake probability of a node being high during periods of increased activity, hence improving the chances of seeing and reporting the event.

4.2.3 Transmission Radius and Detection Threshold

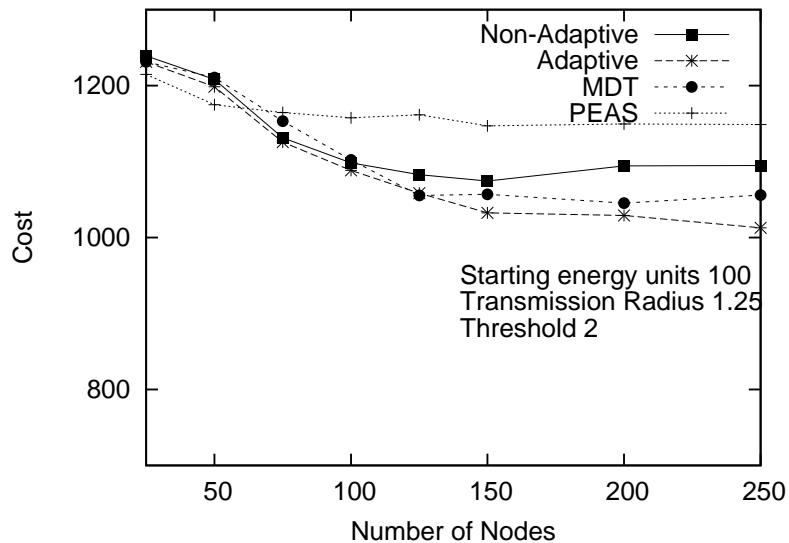
The transmission radius of the nodes plays an important role in routing; the greater the radius, the fewer the number of intermediate nodes used during routing (Figure 4.7(a)) and lesser the energy overhead. Costs increase with increasing node densities due to the effect of false alarms, as explained in 4.2.2. Likewise, a small radius makes the formation of a working network difficult, leading to a degraded performance all round (Figure 4.7(b)) due to increased communication overhead or

even, in the extreme case, nodes being unable to communicate event notifications to the base station. The performance of PEAS improves with increasing transmission radius as the number of nodes awake at any point in time decreases with the radius.

Detection thresholds play an important role in cutting down false alarms. A threshold of 1 (Figure 4.8(a)) leads to a very high rate of false alarms while a threshold of 3 (Figure 4.8(b)) increases the missed event rate without contributing much towards reducing false alarms. A threshold of 2 provides a balance; it cuts down completely on internal false alarms and also keeps missed events and false alarms low.



(a) High Transmission Radius



(b) Low Transmission Radius

Figure 4.7. Effect of transmission radius on cost

4.2.4 Effect of Event Clustering

The adaptive schemes rely on clustering of events for its effectiveness. For adaptation to be effective, there has to be a reasonably good chance that the occurrence of an event signifies that the environmental state is hostile and more events are likely in the near future. This is true in many practical applications, since event arrivals follow Poisson distributions. With the overall average number of events occurring

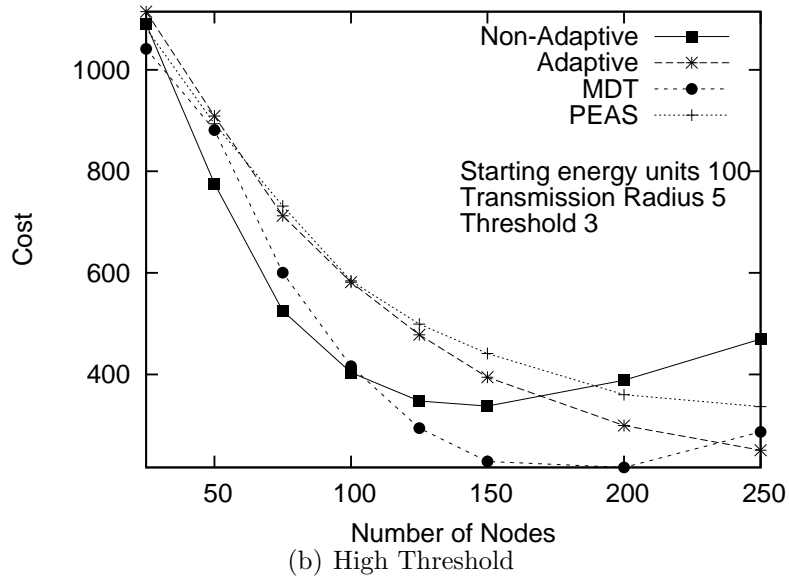
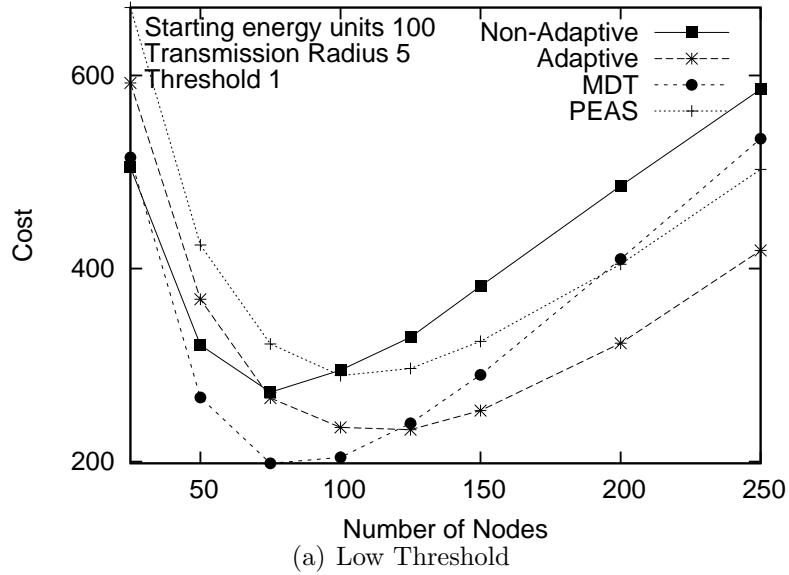


Figure 4.8. Effect of detection threshold on cost

being the same, if there is no clustering, the total amount of time it is awake matters and not the actual wake periods. In this case, the arrival of one event doesn't give us any information as to the arrival of the next event. The performance of the schemes with reduced clustering and no clustering are shown in Figures 4.9(a) and 4.9(b). The aggregate amount of time nodes in the non-adaptive and the simple adaptive schemes stay awake is the same when there is no clustering, hence they see the same number

of events. The MDT scheme also doesn't perform much better or worse as learning is not possible. The average number of events expected in both cases is the same.

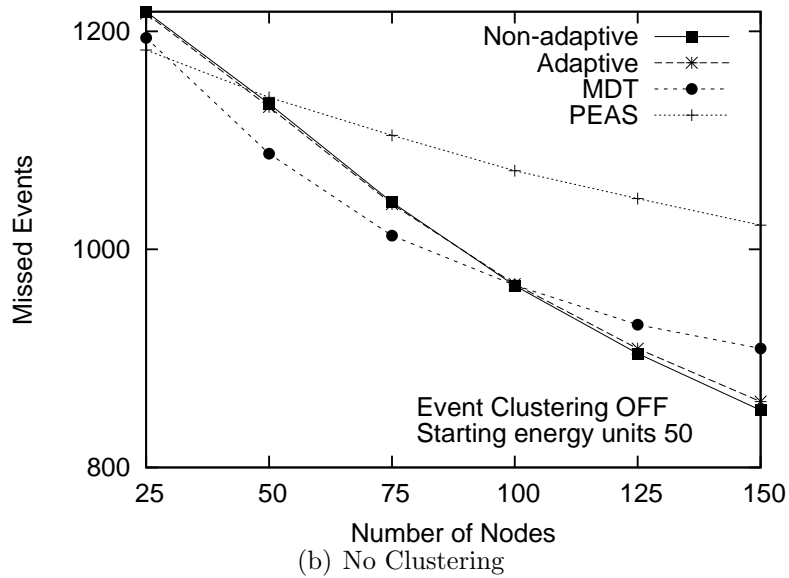
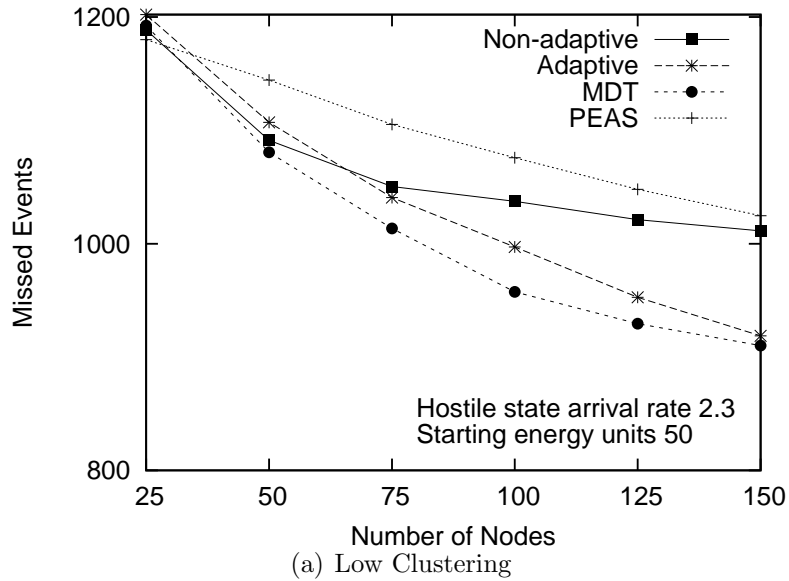


Figure 4.9. Effect of event clustering

4.2.5 Effect of Mission Time

We show in Figures 4.10 and 4.11 the performance of each scheme through its lifetime. Nodes die off much slower in the adaptive scheme, thus maintaining a supe-

rior level of performance throughout the given lifetime. The Simple Adaptive scheme achieves this by waking up minimally, while the MDT scheme cuts out inefficient wake cycles. The MDT scheme is programmed to last the specified mission time. Hence we see a higher rate of node deaths towards the end of the mission time.

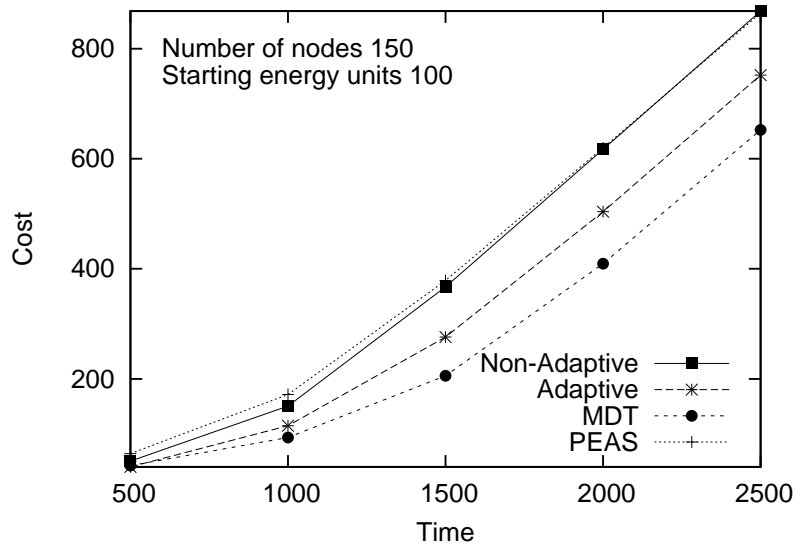


Figure 4.10. Effect of mission time on cost

4.2.6 Duration of Environment

By default, as mentioned earlier, the the sojourn time in each of the environmental states is exponentially distributed. We show the effect of changing the distribution to Erlang in Figure 4.12. We see that there is little difference in the relative performance of the schemes.

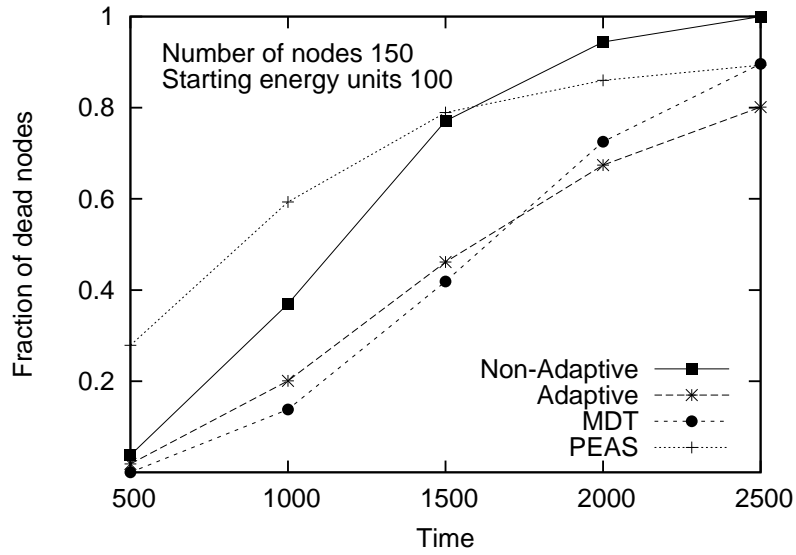


Figure 4.11. Effect of mission time on node lifetime

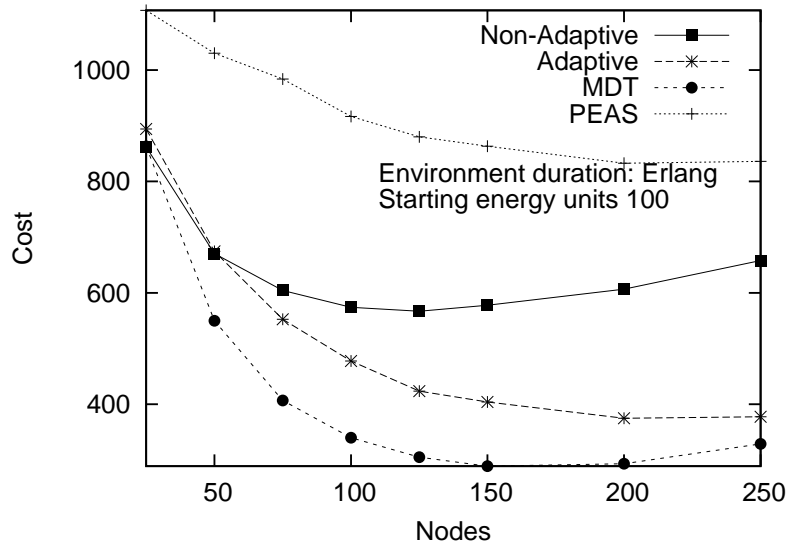


Figure 4.12. Effect of environment duration

4.2.7 Node Distribution

In all the simulations, we use an uniform distribution for nodes. That obviously has an effect on node and system lifetime due to the fact that nodes near the base station have to report the events they see as well as forward reports seen by nodes

on the periphery [21]. To just how much of an effect it has, we distribute nodes such that the node densities decrease exponentially from the center. We see in Figure 4.13 that performance increases substantially especially that of PEAS in comparison to the simple adaptive scheme. The MDT scheme’s performance also increases as the effect of routing, which is not factored into the equations is mitigated.

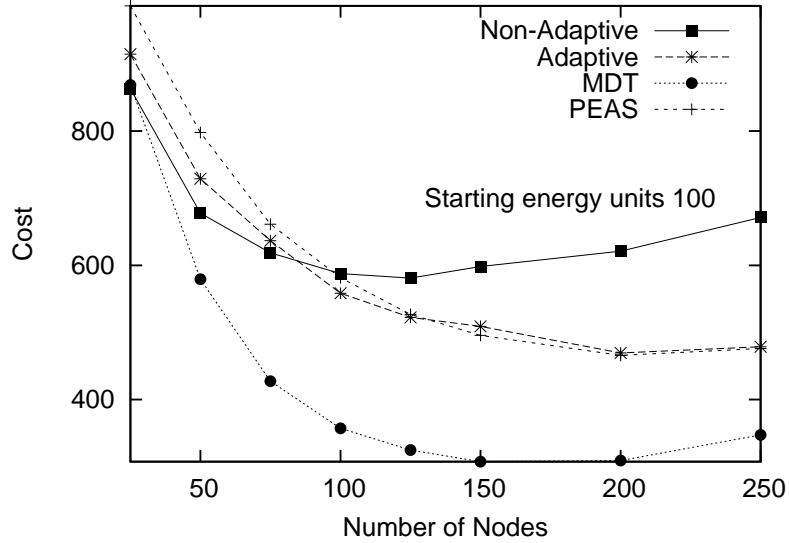
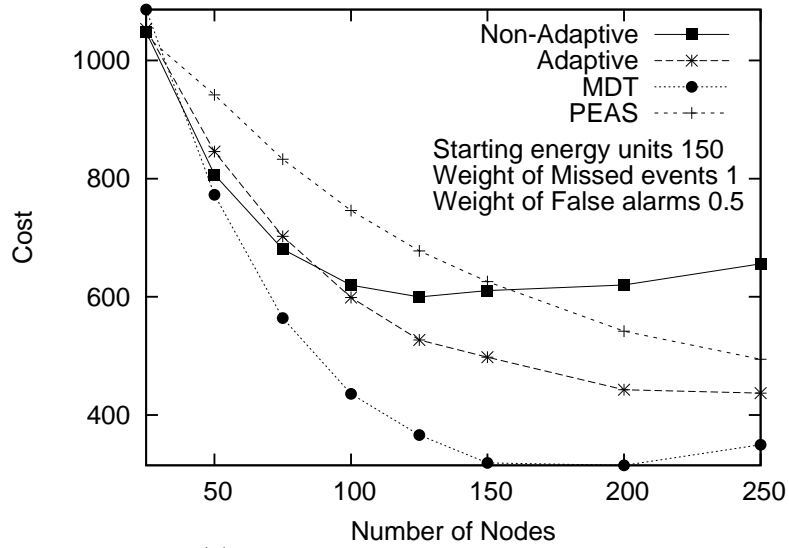


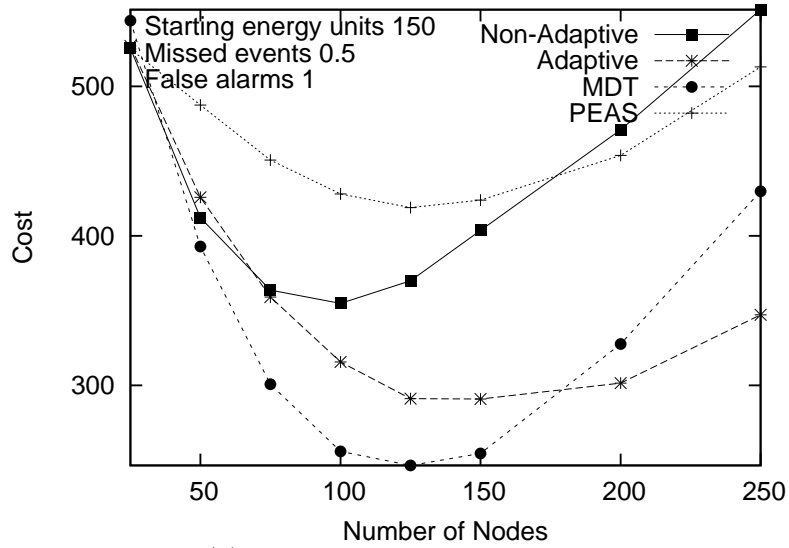
Figure 4.13. Effect of node distribution density

4.2.8 Application-specific requirements

Depending on the application, more weight may be accorded to missed events of false alarms. Figure 4.14(a) shows the cost when missed events are weighted twice as much as false alarms and Figure 4.14(b) shows the cost when missed events are weighted half as much as false alarms. We see that the MDT scheme is very good at reducing missed event cost because it is able to predict and adapt quickly to changing environmental conditions. The simple adaptive scheme is very good at reducing false alarm cost, because it errs more on the side of caution and is liable to miss out on events during transition of the environmental state.



(a) Missed Events More Expensive



(b) False Alarms More Expensive

Figure 4.14. Effect of Application-specific requirements

4.2.9 Benefit of Adaptivity

The performance of the non-adaptive system is heavily dependent on the initial settings. Figures 4.15(a) and 4.15(b) show the effect on performance while Figures 4.16(a) and 4.16(b) show the effect on node lifetime of changing energy and node density settings. The best performance of the non-adaptive system is at different duty cycles for different node densities and starting energy levels, while the adaptive

algorithm is insensitive to these initial settings. Learning is rapid and effective and the system settles on an average duty cycle that gives significantly improved performance. In the static scheme, setting the wake fraction below the optimum value means that a lot of events are missed, while setting it above the optimum level means that the nodes die off sooner.

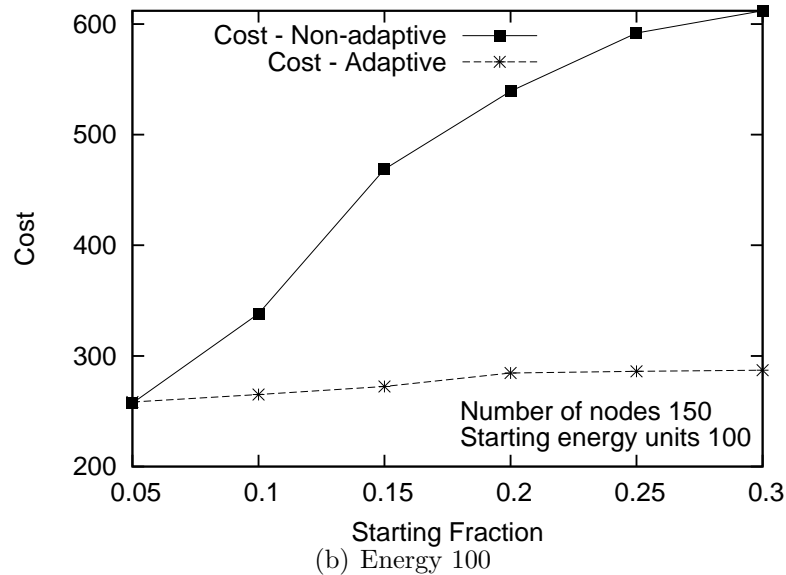
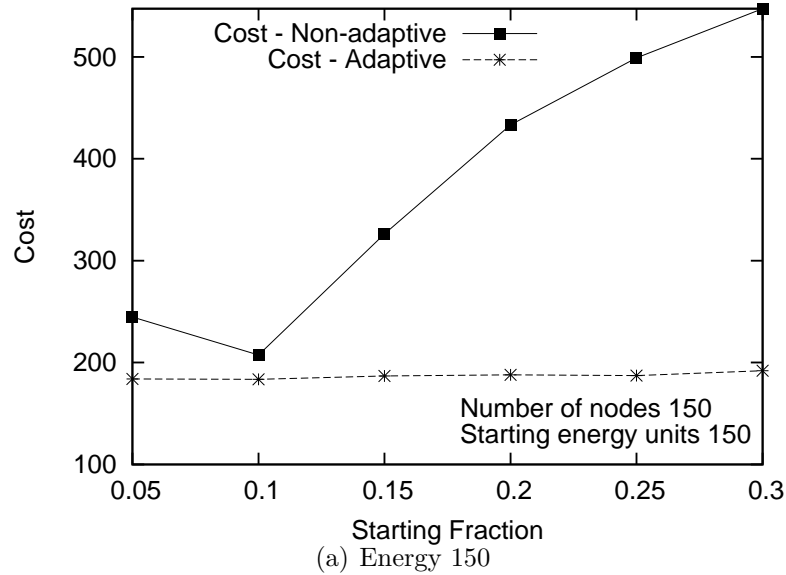


Figure 4.15. Effect of starting wake fraction on cost

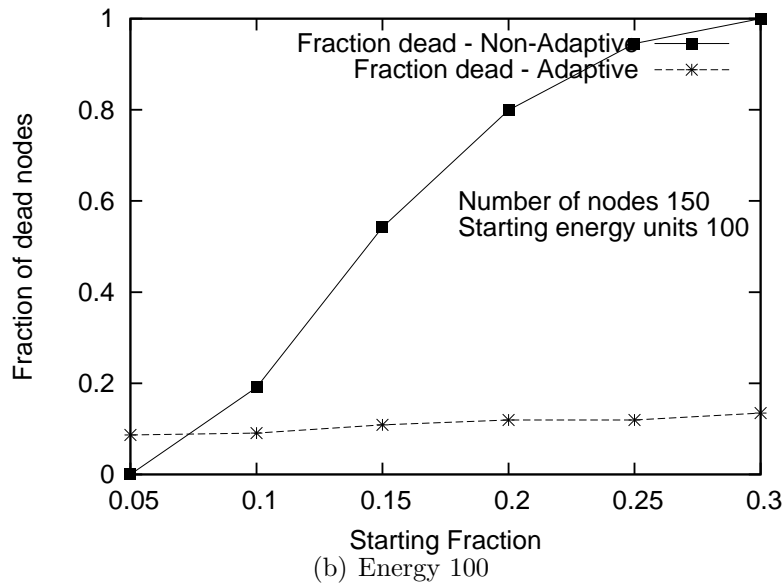
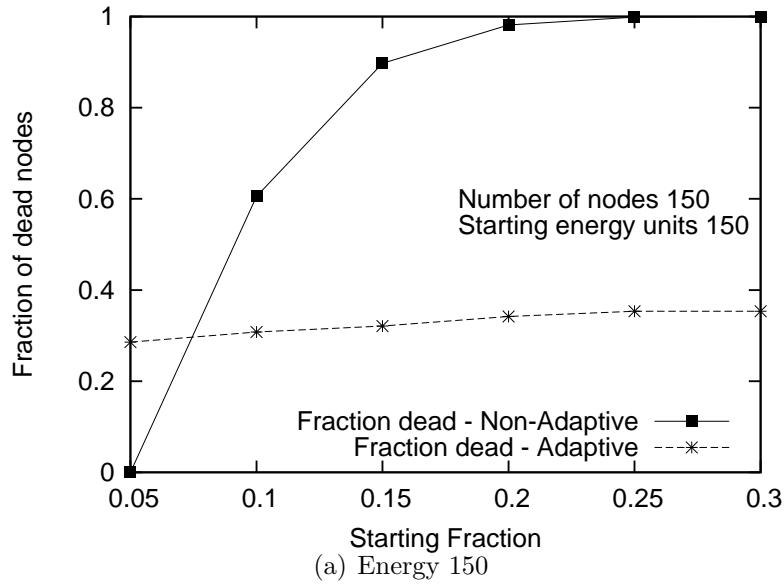


Figure 4.16. Effect of starting wake fraction on node lifetime

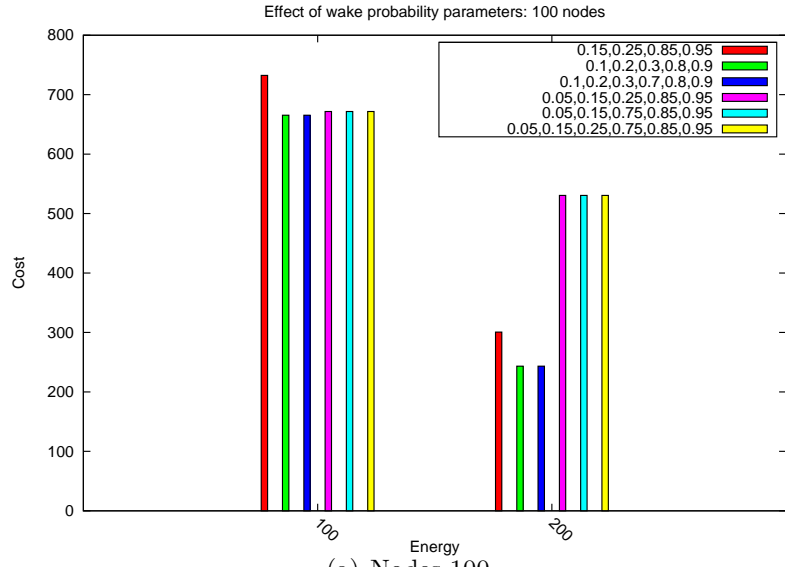
4.2.10 Effect of MDT Parameters

A balance has to be struck between conservatism and aggressiveness with the MDT parameters *i.e.* the probabilities of waking up in the next cycle. The effect is shown in Figures 4.17(a) for 100 nodes and 4.17(b) for 150 nodes. We see that good results can be achieved by a action set of $\{0.1, 0.2, 0.7, 0.8, 0.9\}$, which give the nodes opportunity

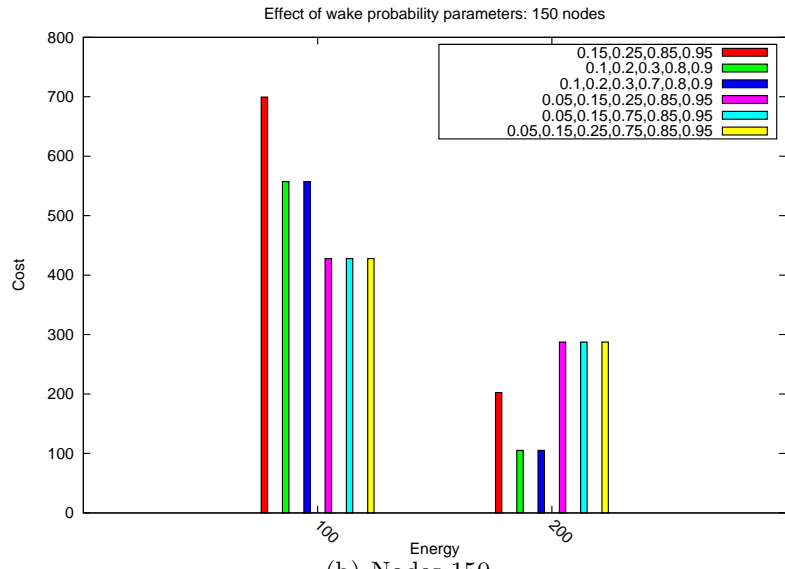
to be conservative when it wants to but also allows it to quickly move to an aggressive state. We also notice that middling probabilities do not affect performance.

Figure 4.18 shows the effect of the minimum wake fraction probabilities. We see that with lower minimum probabilities, cost and false alarms go down for lower energy levels. This is because, with lower minimum wake probability, a node is not forced to wake with high probability when it perceives the external environment to be benign. The rate of adaptation is affected by the wake probabilities, too. With a low minimum probability, if the number of state transitions are not very high (if environment durations are longer) then performance increases as the effect of transitory learning is reduced and systems gets more time to learn and settle. With a higher minimum wake probability, performance decreases because the system doesn't do a good job of conserving energy during the longer benign periods (Figure 4.19). We cannot have too low a minimum value because that would affect network communication by creating blind spots.

Table size for lookup might be an important consideration for sensor nodes with limited memory. Initial table sizes might be of the order of a few megabytes, depending on the mission time and the energy levels. We have seen from experiments (Table 4.2) that compressing the table to a few hundred kilobytes only produces a token hit in performance. The "Compression Factor(T,E)" column denotes the total table compression and the compression applied to the time states and the energy states, respectively. Compression in the time dimension doesn't affect performance as much as energy, because even a small change in the action can throw off the MDT heuristic, especially in constrained systems. This happens slower in the time dimension as energy is a more crucial factor than time and hence faster and more drastically in the energy dimension. We can still get good compression with time states.



(a) Nodes 100



(b) Nodes 150

Figure 4.17. Effect of MDT Parameters

4.2.11 Effect of Rate of Adaptation

For the simple adaptive scheme, the step sizes δ_{inc} and δ_{dec} determine how fast our scheme adapts. The larger the step sizes, the faster the learning. We see from Figures 4.20 and 4.21 that unless the step size is too small, it does not significantly affect performance. It affects the fraction of dead nodes to an extent because larger step sizes mean that in similar time intervals, the amount of time spent awake is

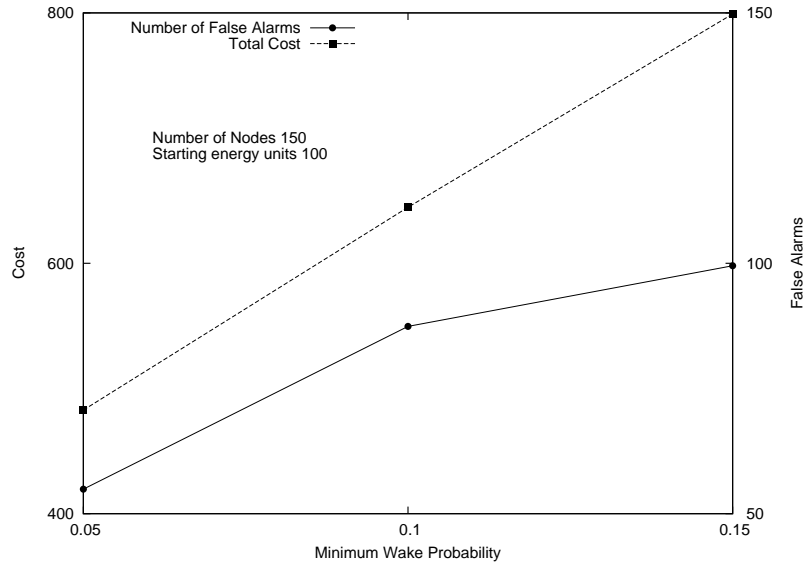


Figure 4.18. Effect of minimum wake fraction on cost

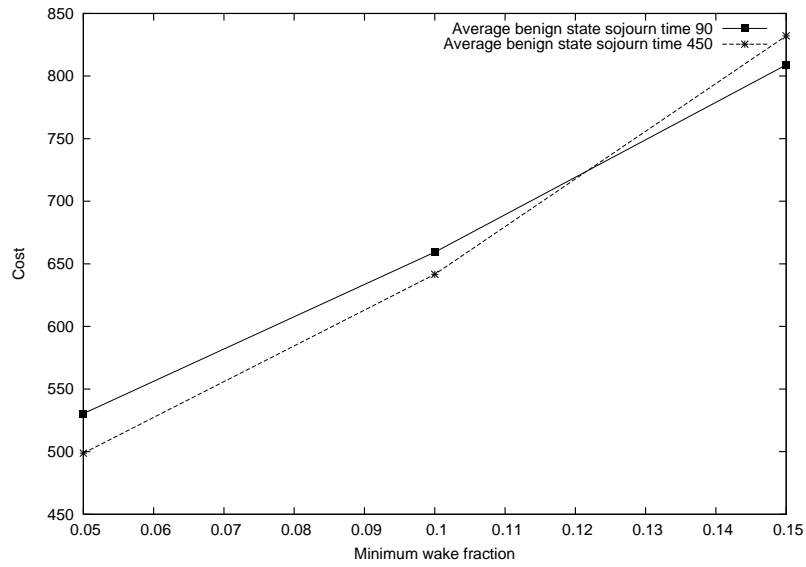


Figure 4.19. Effect of minimum wake fraction on adaptivity

greater. Larger step sizes tend to benefit applications with greater difference in the arrival patterns in the different environments.

Starting Energy 100, Total Nodes 100	
Compression Factor(T,E)	Performance Cost
1(1,1)	709
100(100,1)	711
500(500,1)	720
1250(1250,1)	720
2500(2500,1)	800
2(1,2)	1214
4(1,4)	1215
100(10,10)	1215
1000(100,10)	1260
5000(500,10)	1280

Table 4.2. Effect of Table Compression

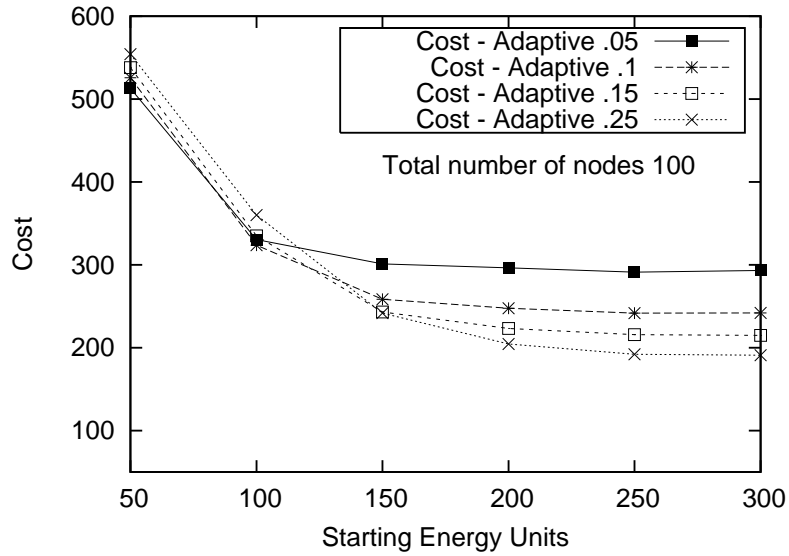


Figure 4.20. Effect of rate of adaptation on cost

4.3 Fault Tolerance

Both the adaptive schemes are inherently fault tolerant. In the simple adaptive, since nodes decide independently on their duty cycles based on event pattern, benign faults do not affect it. If a node dies, its neighbors will carry on regardless. The MDT scheme uses a conservative value of node densities for its calculations. So it too is reasonably robust in that a few nodes dying will not place it at a disadvantage to

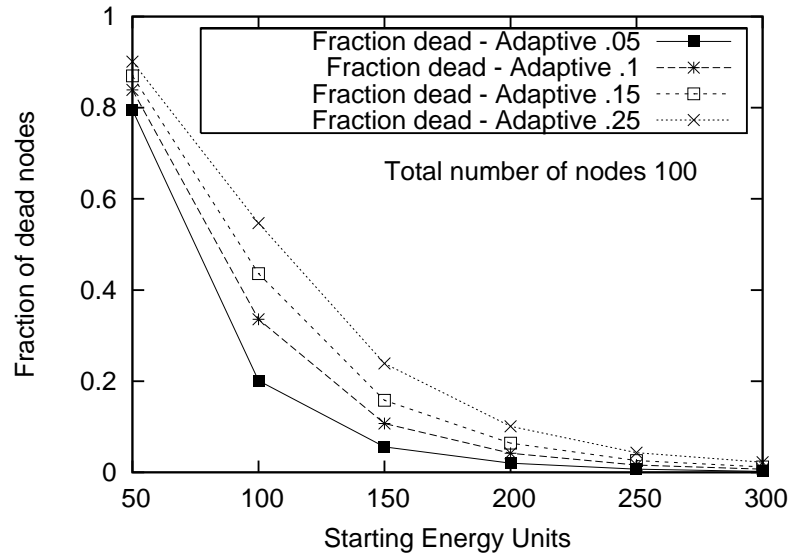


Figure 4.21. Effect of rate of adaptation on node lifetime

other schemes. To be explicitly fault tolerant, all that is required is to use a higher detection threshold. That will make it more robust, but will affect performance cost.

CHAPTER 5

CONCLUSION AND FUTURE WORK

We have presented in this work two adaptive schemes to maximize the performance of an event-detection sensor network under energy constraints. Our schemes technique significantly improves both performance as well as energy efficiency over current schemes. The strength of the schemes lies in their ability to adapt to the characteristics of the monitored event without any specialized additions to hardware. The decision process for the node is very simple, and can be extended to suit the application.

Some of the assumptions and parameter settings that might make an effect on the results are:

- *Communication energy to idle energy ratio:* If communication cost dominates the energy consumption, the benefit obtained from conserving energy might not be as significant, but improvement in cost can still be obtained by having more aggressive step sizes.
- *The duration of the environmental states:* The hostile state has to last long enough for learning to take place. A node which is at its lowest duty cycle might take a handful of cycles in order to reach a level where it can see more events. The duty cycle of the application can be adjusted if we have a rough idea about the environment states' duration.

Possible extensions to the scheme are:

- A node can base its decisions on the number of events seen in a moving window of wake cycles. The benefits of such an extension would depend on whether a longer observation period would give us substantially more information about the environmental state.
- *Randomized* increment (or decrement) of the wake fraction which would allow the node to break localized synchronization of duty cycles which might ensue in the case of deterministic adaptation.
- Energy harvesting could change the decision process significantly. With harvesting, the node only has to worry about extending its lifetime until it is able to recharge itself and that leads to a different decision process. However, interesting trade-offs could occur when a node is low on energy and cannot recharge itself sufficiently in the immediate future.

APPENDIX

THE SIMULATOR

The simulator is written in C but since it also uses data structures from the C++ STL, it requires C++ in order to compile.

A.1 About the Simulator

The simulator simulates a square field of side `$SIDE` with a number of nodes placed based on some distribution. The field is divided into `$SIDE2` cells. Events (real, external noise, internal noise) occur and nodes which sense these events report them to the base station. The base station, depending on the number of reports it gets per event, decides whether an event has occurred or not. The simulator is versatile enough to be able to simulate a variety of scenarios. Its design is modular, and hence can be easily tailored.

The simulator can loop through multiple values of nodes. It can also loop through different starting energy levels for all schemes except the MDT. The simulation results are saved in `$HOME/out/nodes$i`, where `$HOME` is the simulator's home directory, and `$i` is the value `num_nodes` in table A.1. The output file reports the number of missed events, false alarms, the performance cost and the fraction of nodes surviving the simulation time, for each energy level and node densities. The values are reported for node threshold values of 1 through `THRES_MAX`.

To implement a new scheme, a subroutine `nodeon()` needs to be implemented, so that it is compiled when the `ALGO` macro is set to the new scheme. To change the routing scheme, the `routemsg()` subroutine can be swapped. To change environmental

patterns, the template in function `state1amb()` can be followed, where exponential and uniform sojourn times are implemented. To change the node distribution pattern, the template in function `insert()` can be followed, where uniform and exponential density distributions are implemented.

A.2 Installation and Compilation

The tarball can be downloaded here:

<http://arts.ecs.umass.edu/~ssundare/simulator/simulator.tgz>. The simulator root directory contains all the required directory trees. For the Base, Simple Adaptive and the PEAS schemes, a simple `make` would compile the code. For the MDT scheme, “SCH= MDTx” has to be specified with `make`, where “x” is “W” or “R”. The MDT scheme may be run in two steps. The first step is to create the lookup tables. For that purpose, the option used is `MDTW`. To run the simulations using the created lookup tables, `MDTR` needs to be used. The reason why this has been made a two-step process is that it takes a long time to create tables. Tables are created in `out/tables` for each energy level and node density. Tables created for one set of parameters can be reused at a later point. To run it as a single process, the option “OPT” in the Makefile should be overridden to `NULL`. To reuse tables in different directories, the file open statement in `src/tableread.c` can be modified. *The object files, created in the `obj/` directory need to be cleaned while switching between the MDT Read and Write modes.* The `make clean` command cleans the object files. `make cleanall` cleans the executables in the `bin/` directory as well. The executable is created in the `bin/` directory.

[†]For Non-MDT schemes

[‡]For Non-MDT schemes it is specified by the macro `ENERGY_int`

A.3 Examples

- To make and run a non-MDP scheme (after changing the `ALGO` macro in `include/head.h`:

1. `make`
2. `./bin/sim`

- To make and run an MDP scheme to just create tables:

1. `make SCH='MDTW'`
2. `./bin/sim`

- To make and run an MDP scheme to read tables and simulate the system:

1. `make SCH='MDTR'`
2. `./bin/sim`

- To make and run an MDP scheme in one go:

1. `make SCH='MDTR' OPT=`
2. `./bin/sim`

- To add debug options for any scheme:

1. `make DBG=-ggdb`
2. `./bin/sim`

Parameter	Occurrence	Description
SIDE	macro	Field size
DUTY	macro	Unit of time
TIME	macro	Period Duty Cycle
num_nodes	main()	Node count multiplier
N_N	main()	Number of nodes (default: $num_nodes \times 25$)
ener	main()	Starting Energy count multiplier [†]
ENERGY	main()	Starting Energy (default: $ener \times 50$) [‡]
T	main()	Simulation time
ALGO	macro	Enumerated variable for scheme used
THETA	main()	Signal detection threshold for nodes
ALPHA	main()	Signal attenuation power
LAMBDA_E_H	macro	Event arrival rate in hostile environment
LAMBDA_E_B	macro	Event arrival rate in benign environment
LAMBDA_N	main()	Overall noise arrival rate
LAMBDA_EXT_N	main()	External noise arrival rate
LAMBDA_INT_N	main()	Internal noise arrival rate
BEN_TO_HOS	macro	Rate of change of environment from benign to hostile
HOS_TO_BEN	macro	Rate of change of environment from hostile to benign
TRANS_EN	macro	Energy consumed by node by transmitting one message
REC_EN	macro	Energy consumed by node by receiving one message
IDLE_EN	macro	Energy consumed by node during idling
SIG_MEAN	macro	Mean signal strength for real events
NOISE	macro	Mean signal strength for noise events
COST_ME	macro	Cost of missed events
COST_FA	macro	Cost of false alarms
OUTSTEPS	macro	The time slices after which results are saved
ENV_HOLDING_TIME	macro	Distribution used for environment changes
TRANS_RAD	macro	Transmission radius for nodes
BASE_[X,Y]	macro	Co-ordinates of Base station
NODE_DIST	macro	Distribution used for node density distribution
DEC_DELFRAC	main()	Fraction by which Duty cycle is decremented in Simple Adaptive
INC_DELFRAC	main()	Fraction by which Duty cycle is incremented in Simple Adaptive
MINFRAC	main()	Lower bound fraction to which duty cycle can be reduced in Simple Adaptive
MAXFRAC	main()	Upper bound fraction to which duty cycle can be reduced in Simple Adaptive
PEAS_MEAN_TIME_TO_WAKE	macro	Mean duration at which nodes implementing PEAS wake up

Table A.1. Parameter Names

Data Structures	Description
node	The node data structure; contains information pertaining to energy level, duty cycling, position, <i>etc</i>
nodelist[]	Array of generated nodes
head[][]	List of nodes in each cell
event	The event data structure; the time and location of occurrence
event_msg	The structure used in the routing algorithm
Table[][][]	The uncompressed lookup table
TableComp[][][]	The compressed lookup table
Ph[]	The set of probable environmental states
Act[]	The set of actions for a node

Table A.2. Data Structures

Function	Description
createlist	Creates list of nodes
getrand	Generates a uniform random number in $(0, 1)$ using Mersenne Twister
event_occur	Generates event
response	Returns the number of nodes that saw a particular event
nodeon_base	Non-adaptive scheme
nodeon_duty	Simple Adaptive scheme
nodeon_mdp	MDT scheme
nodeon_peas	PEAS scheme
countdead.T	Returns the number of nodes that are dead after time T
routemsg	Forwards a message to base-station

Table A.3. Important functions

BIBLIOGRAPHY

- [1] Arora, A., Dutta, P., Bapat, S., Kulathumani, V., Zhang, H., Naik, V., Mittal, V., Cao, H., Demirbas, M., Gouda, M., Choi, Y., Herman, T., Kulkarni, S., Arumugam, U., Nesterenko, M., Vora, A., and Miyashita, M. A line in the sand: a wireless sensor network for target detection, classification, and tracking. *Computer Networks* 46, 5 (2004), 605–634.
- [2] Buettner, M., Yee, G.V., Anderson, E., and Han, R. X-MAC: A short preamble MAC protocol for duty-cycled wireless sensor networks. In *Proceedings of the 3rd International Conference on Embedded Networked Systems (Sensys)* (2006), pp. 307–320.
- [3] Chiasserini, C.F., and Garetto, M. Modeling the performance of wireless sensor networks. In *Proceedings of the Annual IEEE Conference on Computer Communications (INFOCOM)* (2004), pp. 220–231.
- [4] Dubois-Ferriere, H., Estrin, D., and Vetterli, M. Packet combining in sensor networks. In *Proceedings of the 3rd ACM Conference on Embedded Networked Systems (Sensys)* (2005), pp. 102–115.
- [5] Ganeriwal, S., Ganesan, D., Sim, H., Tsiatsis, V., and Srivastava, M. Estimating clock uncertainty for efficient duty-cycling in sensor networks. In *Proceedings of the 3rd International Conference on Embedded Networked Systems (SenSys)* (2005), pp. 130–141.
- [6] Ghosh, A., and Givargis, T. Lord: A localized, reactive and distributed protocol for node scheduling in wireless sensor networks. In *Design, Automation and Test in Europe (DATE), Volume 1* (2005), pp. 190–195.
- [7] Gu, L., and Stankovic, J. Radio-triggered wake-up capability for sensor networks. In *Proceedings of the 10th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)* (2004), p. 27.
- [8] Karp, B., and Kung, H.T. GPSR: Greedy perimeter stateless routing for wireless networks. In *Proceedings of the 6th Annual International Conference on Mobile computing and networking (MobiCom)* (2000), pp. 243–254.
- [9] Liu, Juan, Koutsoukos, Xenofon, James, and Zhao, Reich Feng. Sensing field: Coverage characterization in distributed sensor networks. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* (2003), pp. 173–176.

- [10] Mainwaring, A., Polastre, J., Szewczyk, R., Culler, D., and Anderson, J. Wireless sensor networks for habitat monitoring. In *Proceedings of the First ACM International Workshop on Wireless Sensor Networks and Applications (WSNA)* (2002), pp. 88–97.
- [11] Maroti, M., Kusy, B., Balogh, G., Volgyesi, P., Nadas, A., Molnar, K., Dora, S., and Ledeczi, A. Radio interferometric geolocation. In *Proceedings of the 3rd ACM Conference on Embedded Networked Systems (Sensys)* (2005), pp. 1–12.
- [12] P. Dutta, M. Grimmer, Arora, A., Bibyk, S., and Culler, D. Design of a wireless sensor network platform for detecting rare, random and ephemeral events. In *Fourth International Symposium on Information Processing in Sensor Networks (IPSN)* (2005), pp. 495–502.
- [13] Patwari, N., and Hero, A.O. Indirect radio interferometric localization via pairwise distances. In *Third IEEE Conf on Embedded Sensor Networks (EmNets)* (2006).
- [14] Polastre, Joseph, Hill, Jason, and Culler, David. Versatile low power media access for wireless sensor networks. In *Proceedings of the 2nd international conference on Embedded networked sensor systems (SenSys)* (2004), pp. 95–107.
- [15] Savvides, A., Park, H., and Srivastava, M.B. The bits and flops of the n-hop multilateration primitive for node localization problems. In *Proceedings of the First ACM international workshop on Wireless sensor networks and applications (WSNA)* (2002), pp. 112–121.
- [16] Sinha, A., Wang, A., and Chandrakasan, A. Algorithmic transforms for efficient energy scalable computation. In *Proceedings of the 2000 International Symposium on Low Power Electronics and Design (ISLPED)* (2000), pp. 31–36.
- [17] Sinha, Amit, and Chandrakasan, Anantha. Dynamic power management in wireless sensor networks. *IEEE Design and Test of Computers* 18, 2 (March 2001), 62–74.
- [18] Stojmenovic, Ivan, Ed. *Handbook of Sensor Networks*. Wiley, 2005, ch. 1, pp. i–xxxviii.
- [19] van Dam, Tijs, and Langendoen, Koen. An adaptive energy-efficient mac protocol for wireless sensor networks. In *Proceedings of the 1st international conference on Embedded networked sensor systems (SenSys)* (2003), pp. 171–180.
- [20] Werner-Allen, G., Lorincz, K., Welsh, M., Marcillo, O., Johnson, J., Ruiz, M., and Lees, J. Deploying a wireless sensor network on an active volcano. *IEEE Internet Computing* 10, 2 (2006), 18–25.
- [21] Xue, Q., and Ganz, A. On the lifetime of large scale sensor networks. *Elsevier Computer Communications* 29, 4 (2006), 502–510.

- [22] Ye, F., Zhong, G., Lu, S., and Zhang, L. Peas: A robust energy conserving protocol for long-lived sensor networks. In *International Conference on Distributed Computing Systems (ICDCS)* (2003), pp. 28–37.
- [23] Ye, W., Heidemann, J., and Estrin, D. An energy-efficient mac protocol for wireless sensor networks. In *Proceedings of the Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)* (2002), pp. 1567–1576.
- [24] Zhao, Jerry, and Govindan, Ramesh. Understanding packet delivery performance in dense wireless sensor networks. In *Proceedings of the 1st international conference on Embedded networked sensor systems (SenSys)* (2003), pp. 1–13.