Adaptive Sleep Scheduling for Energy-efficient Movement-predicted Wireless Communication *

Yu Dong and David K. Y. Yau
Department of Computer Science, Purdue University, West Lafayette, IN 47906
{dong, yau}@cs.purdue.edu

Abstract

Energy efficiency in network communication is critical for wirelessly connected small computing devices, which run on limited battery capacity. Under realistic movement scenarios (e.g., a person traveling at airplane, automobile, or biking speed), a mobile sender can track its own movement and postpone communication (subject to application deadline constraints) until it moves close to the communication target. This will save significant energy of sending, which grows superlinearly with the communication distance in, say, the single hop wireless context. However, movement tracking requires the mobile device to be turned on and hence consumes energy. Instead of continuous tracking, the mobile device should sample its movement and be allowed to sleep between the sampling instants (provided that the application also does not have work to do during the sleep). In this paper, we present an adaptive scheduler for determining an effective sampling schedule given changing operating conditions. Our experimental results show that the scheduler can achieve substantial energy savings over a device that is always on. Moreover, the scheduler's adaptivity allows it to outperform fixed sleep periods between tracking, since the "right" sleep period depends on dynamic system conditions and cannot be determined a priori.

1. Introduction

Current progress in computing technologies makes it possible to build embedded systems on tiny computing devices like sensors, on which diverse applications can be developed. However, these tiny sensor devices are typically resource-limited: lower computation capacity, smaller memory for computing, shorter radio range of communication, and most of all, limited power supply. The limited power supply requires efficient energy use strategies such that the sensor devices can complete their assigned tasks within the energy budget.

To increase the lifetime of power-limited mobile devices, much research has focused on energy efficiency by using either hardware or software approaches to reduce the energy consumption on these devices. Wireless network communication can be energy expensive, and should be targeted for significant energy savings. Previous work [3] has shown how the movement history of a mobile sender can be used to predict if the sender will likely move close to its receiver by an application deadline. If so, the data transmission can be postponed until the predicted time when the communicating parties are close to each other, thereby saving transmission energy. In the case of single hop communication, the energy saving is obvious since the required transmission power grows superlinearly in the transmission distance. In the case of multihop transmission, reduced physical distance between the two nodes does not necessarily imply a reduction in the network distance (i.e., the number of hops between the two nodes). However, reduction in the network distance is still highly *likely* provided that the network is not very sparse. Moreover, we can expect the lengths of the individual hops to be smaller.

Whereas motion prediction can be useful, its performance depends on the accuracy of movement tracking. Continuous tracking would require a mobile device to be always turned on, which can consume excessive energy. Instead of continuous tracking, a mobile device should sample its movement and be allowed to sleep between the sampling instants, provided that the application also does not have work to do between the two instants. We will show that more frequent position sampling will generally lead to better expected performance of motion prediction, which will in turn lead to reduced energy use of wireless transmission. Conversely, however, less frequent sampling will allow longer sleep periods, and hence better energy conservation of the idle device, between samples. We must therefore balance between the dual goals of saving energy for network transmission versus saving energy for the overall

*Research was supported in part by the National Science Foundation under grant numbers CCR-9875742 (CAREER) and CNS-0305496, and in part by an IBM Fellowship awarded to Y. Dong.

system. An *adaptive scheduler* should be designed for a mobile sender to determine an *optimal* frequency of sampling (and hence, a suitable packet transmission schedule) based on dynamic operating conditions.

We have evaluated the performance of our adaptive scheduler through both packet-level network simulations and detailed measurements of a prototype implementation on the Berkeley mote. Our experimental evaluations are driven by several real world movement scenarios of a mobile device – e.g., a device carried by a walking person, a runner, a bicycle, an automobile, or an airplane. Our results show that the scheduler can simultaneously reduce the energy use by the wireless communication and by the total system, thereby significantly extending the lifetime of mobile sensor devices.

1.1. Paper organization

The balance of the paper is organized as follows. Necessary background of energy-efficient motion-predicted wireless communication is presented in Section 2. The system model is defined, and the need for on-line determination of suitable sleep times is motivated. In Section 3, we propose four adaptive sleep schedulers for computing effective sleep and position update schedules under changing system conditions. Section 4 discusses the design and implementation of our power management architecture on a TinyOS/Berkeley mote device. Simulation and implementation results and their analysis are presented in Section 5. Related work is discussed in Section 6. Section 7 concludes.

2. Sampling Schedule of Movement Prediction

2.1. Basics of movement-predicted wireless communication

We review the prior work in [3] on movement-predicted energy-efficient wireless communication.

2.1.1 System Model

We consider the system model of an ad-hoc wireless network in which a set of mobile nodes communicate with a stationary receiver node, which we call the *target*. (A stationary target simplifies discussions, but the assumption can be easily removed to address a mobile target, as discussed in [3].) The whole network is divided into virtual grids in two-dimensional space. (Generalization to 3D space is straightforward.) We assume that each node knows its position through GPS (or similar positioning systems) and consequently can associate itself with a grid. We assume slotted time and that a node remembers its movement history as the sequence of grid IDs it visited during the previous n time slots. We also assume that all the nodes know the position y of the target.

There are two main system parameters used by a mobile node to predict its movement: The length n of the movement history remembered, and the maximum allowable delay, denoted as D (in seconds), for which a communication can be postponed since arrival. We also define the following terms. Let N be the total number of grids in the network. $S_h = \{x_1, x_2, ..., x_m\}$ is the sequence of the m previous grid positions visited by the mobile node h, where x_i is the ith grid ID and x_m is the most recent grid ID visited in the sequence. We also define $d(i, j)$ as the Euclidean distance between two grid positions i and j.

2.1.2 Postponement algorithm

To reduce energy use in wireless transmission, a mobile sender attempts to postpone communication until the predicted time when it moves "close" to the target node, subject to the application deadline D. When the deadline is reached, the node will carry out the communication at the Dth time slot without waiting any further. At this time, if the mobile node is further from the target than it was, it will likely end up using more energy and we call this the *estimation penalty*.

Several postponement algorithms are studied and extensively evaluated in [3]. Among them, the *Least Distance* (LD) algorithm is shown to be the simplest and most effective. Unless otherwise specified, we use LD as the postponement algorithm in this paper. We now review its essential properties.

The LD algorithm is based on the 37% rule of the Best-choice(r) algorithm, which solves the well known secretary problem. The secretary problem presents a set of candidates sequentially. When a candidate is presented, an irrevocable choice must be made either to accept or reject the candidate. It is thus similar to the sequence of decisions a mobile sender will have to go through deciding whether to communicate or not. According to the 37% rule, the first 37% of the candidates are just evaluated, but not accepted. Then we take the candidate whose relative rank is the first among the candidates seen so far [6]. In our case, we assume that we have already seen the first 37% or more of the candidate positions as the location history. We first find the least distance d_{min} between a mobile node and the target in the history of that node:

$$d_{min} = Min_{\forall x \in S_h} d(x, y)$$

Then, in each of the next D time slots we check if the current distance d is less than or equal to d_{min}. At any time slot, if we find $d \leq d_{min}$, we communicate immediately; else we communicate at the Dth time slot.

2.2. How position sampling affects movement prediction

In [3], a constant position sampling rate of one sample/time slot is assumed. In general, the sampling rate can be considered a system parameter denoted by R (in samples/second). Then $k = DR$ gives the maximum

number of samples that can be obtained by the deadline. We now illustrate how R can affect the performance of a postponement algorithm, given a simplistic uniform random mobility model of how a node moves (i.e., the mobile node is equally likely to move to any grid position in each time slot). The postponement algorithm is such that the mobile sender will decide to communicate once it is tracked to be in the same grid position as the target. Otherwise, communication is postponed until the deadline is reached.

Theorem 1 *Given a maximum allowable delay D, the probability that a mobile node is not tracked to be in the grid of the target node at sampling rate R_1 is strictly lower than the same probability at a lower sampling rate R_2.*

Proof: Please refer to [5]. □

Theorem 1 tells us that a mobile node using a higher position sampling rate is always less likely to miss a close position for communicating with the target. On the basis of Theorem 1, we show how R may affect the communication energy saving achieved by the postponement algorithm. We let y be the target's position, C_i be the energy cost of network communication if the mobile node is in grid ID $i, (1 \leq i \leq N)$, and C'_y be the expected energy cost of network communication if the mobile node is not in grid ID y. Now,

$$C'_y = \sum_{i=1, i \neq y}^{N} \frac{1}{N-1} C_i \tag{1}$$

The expected energy cost of communication without postponement is

$$E_o = C'_y \tag{2}$$

and the expected energy cost with postponement is

$$E_m = [1 - (1 - \frac{1}{N})^k] C_y + (1 - \frac{1}{N})^k C'_y \tag{3}$$

The expected energy saving is

$$E_g = E_o - E_m = [1 - (1 - \frac{1}{N})^k](C'_y - C_y) \tag{4}$$

The next theorem shows how R may affect the expected energy saving, denoted by E_g, of the wireless communication.

Theorem 2 *Given a maximum allowable delay D, the postponement algorithm can achieve a strictly higher expected energy saving E_g on the communication when the position sampling rate is higher.*

Proof: Please refer to [5]. □

2.3. Sleep more or sample more?

Theorems 1 and 2 suggest that, by applying the postponement algorithm, a mobile node should sample its position information as frequently as possible to maximize the energy saving of the communication. The situation can be illustrated in Figure 1(a), which shows the network energy consumption as a function of the sleep period between two consecutive samplings, denoted by t_{sleep}. We expect the function to be monotonically increasing, since longer sleeps will lead to less accurate tracking and also more misses of good opportunities for communication.

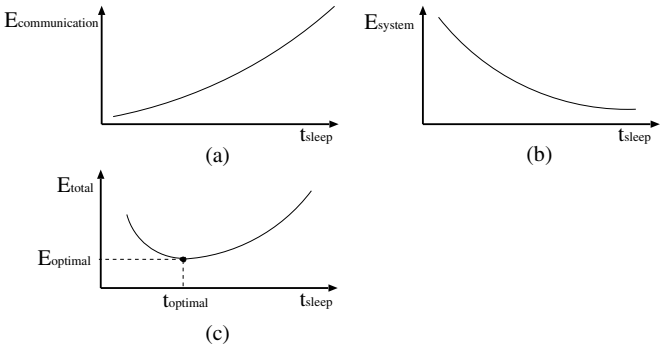


Figure 1. Energy consumption as a function of the fixed sleep period t_{sleep}, for (a) communication, (b) system without communication, and (c) overall system.

On the other hand, a sleeping device consumes minimal energy, and a longer sleep period will cause less frequent transitions between the on/off states of the device. Hence, if we ignore the need for network communication, energy use of the overall system will decrease as the sleep period increases. This is illustrated by the monotonically decreasing function in Fig. 1(b). Jointly considering the two sources of energy expenditure in the total system, Fig. 1(c) shows the inherent tradeoff between saving more energy of wireless communication by sleeping less often and saving more energy of the overall system by sleeping more often. Notice from the figure that there exists an optimal sleep period, denoted by $t_{optimal}$, at which the total energy use by the system is minimized.

In practice, determining $t_{optimal}$ can be difficult, since its value changes according to dynamic conditions such as the current movement of the mobile device and the remaining delay budget of the application. We now discuss how an *adaptive* scheduler can be designed to track the value of $t_{optimal}$ in the presence of such system dynamics.

3. Adaptive Wakeup Scheduling

Instead of using a fixed sleep period between samples, a mobile device should adapt its sleep schedule online. Specifically, each time the device wakes up to sample its $(i-1)$st position, it also determines the sleep time until the ith position update based on the current system parameters.

We have studied four adaptation strategies: *speed-based adaptation* (SBA), *delay budget-based adaptation* (DBA), *prediction performance-based adaptation* (PePA), and *position-based adaptation* (PoBA). SBA and PoBA adapt the sleep period to the nodal movement, DBA adapts based on the current remaining delay budget, while PePA adapts based on the observed achievable distance saving so far. SBA and PoBA work by assuming that the nodal speed and direction will likely not change a lot during the sleep provided that the sleep period is reasonably small in the context of the nodal movement. E.g., a cruising airplane will likely not change its speed and direction much over say 30 minutes, while a running athlete will likely maintain his current movement over a period of say 30 seconds.

3.1. Speed-based adaptation (SBA)

When a mobile device moves at a higher (e.g., automobile) speed, it may require more frequent position updates to prevent missing significant position changes which may affect the decision to communicate or not. On the other hand, a mobile device moving at low speed (e.g., a walking person) can afford to increase the sleep period without missing critical information.

SBA uses the above observation to adapt the sleep period to the current nodal speed observed. Also, since our movement model is based on a grid-based network, we can determine the sleep time until the next position update as follows:

$$Speed_i = \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2}/T_{sleep_{i-1}} \quad (5)$$

$$T_{sleep_i} = (\frac{Side}{2})/Speed_i \quad (6)$$

where T_{sleep_i} is the sleep time to use until the ith (i.e., next) position update, $Side$ is the length of a grid, and $Speed_i$ is the currently observed nodal speed. Physically, Equation 6 calculates the expected time required for the node to move to a neighbor grid given its current speed. This time is then used as the sleep time until the next position update. Let MP denote LD motion prediction algorithm hereafter, and the pseudo-code of SBA is given as follows:

```
SPEED_BASED_ADAPTATION()
 1  At each wakeup
 2    Update position information
 3    Calculate current speed_i
 4    Calculate T_sleep_i
 5    for each data packet in Queue
 6      if MP predicts current position is good or deadline reaches
 7        or delay + T_sleep_i > deadline
 8          send packet
 9      else
10        delay = delay + T_sleep_i
11      end if
12    end for
13    SLEEP(T_sleep_i)
14  end wakeup
```

3.2. Delay budget-based adaptation (DBA)

When the communication deadline is fast approaching (i.e., little remaining delay budget before communication is due), a mobile device may require more attempts to find a closer position to the target before the deadline expires. Thus, more frequent position updates are suggested, and a shorter sleep time should be used. Accordingly, DBA adapts the sleep period to the average delay budget of the pending communications (i.e., the buffered packets waiting to be sent). The sleep time until the ith (i.e., next) position update is calculated as follows:

$$r_{delay_i} = (\frac{1}{M}\sum_{i=1}^{M} d_i)/D \quad (7)$$

$$T_{sleep_i} = (1 - r_{delay_i}) \cdot \alpha \cdot D \quad (8)$$

where M is the total number of pending communications, D is the maximum allowable delay for each communication (Equation 8), α is a tunable system parameter that adjusts the weight of the delay budget ($(1-r_{delay_i}) \cdot D$) in calculating the next sleep period. A larger α will allow more sleep until the deadline is reached. The performance results in Section 5 show how α can affect the sleep period and hence the performance of adaptation. The pseudo-code of DBA is given as follows:

```
DELAY_BUDGET_BASED_ADAPTATION(α)
 1  At each wakeup
 2    Update position information
 3    for each data packet in Queue
 4      Calculate remaining delay budget
 5    end for
 6    Calculate T_sleep_i with α
 7    for each data packet in Queue
 8      if MP predicts current position is good or deadline reaches
 9        or delay + T_sleep_i > deadline
10          send packet
11      else
12        delay = delay + T_sleep_i
13      end if
14    end for
15    SLEEP(T_sleep_i)
16  end wakeup
```

3.3. Prediction performance-based adaptation (PeBA)

An important performance metric of movement prediction is the saving in communication distance achieved so far, which can be observed in real time. With PeBA, we use the distance saving information to determine the sleep time until the next update. The distance saving achieved at the current update is compared with the last saving achieved.

If the distance saving decreases, the mobile device should be given more attempts to find a good position to communicate, by sampling more frequently. Otherwise, the sleep time can increase. Specifically, the sleep time is calculated as follows:

$$r_{save_i} = (\frac{1}{M}\sum_{j=1}^{M} dist_{c_j})/(\frac{1}{M}\sum_{j=1}^{M} dist_{o_j}) \quad (9)$$

$$T_{sleep_i} = (r_{save_{i-1}}/r_{save_i}) \cdot T_{sleep_{i-1}} \quad (10)$$

where $dist_{o_j}$ is the communication distance of the jth pending packet from the target at the time the packet was generated ($1 \leq j \leq M$, for M pending communications), and $dist_{c_j}$ is the current communication distance of the jth pending packet from the target. r_{save_i} is thus the ratio of the average current distance to the average original distance. T_{sleep_i} is the sleep time until the next position update. The pseudo-code of the PeBA is given as follows:

```
PREDICTION_PERFORMANCE_BASED_ADAPTATION()
 1  At each wakeup
 2    Update position information
 3    for each data packet in Queue
 4      Calculate current distance saving
 5    end for
 6    Calculate T_sleep_i with α
 7    for each data packet in Queue
 8      if MP predicts current position is good or deadline reaches
 9        or delay + T_sleep_i > deadline
10        send packet
11      else
12        delay = delay + T_sleep_i
13      end if
14    end for
15    SLEEP(T_sleep_i)
16  end wakeup
```

3.4. Position-based adaptation (PoBA)

We can combine the information used in SBA (i.e., nodal speed) and DBA (i.e., remaining delay budget) to effect a position-based adaptation strategy (PoBA). With PoBA, the node, on waking up, tries to determine its sleep time T_{sleep_i} until the next position update. To do so, let $delay_{max}$ be the longest time by which any of the node's pending communications has been delayed, so that $D' = D - delay_{max}$ is the smallest delay budget among the pending communications. Iteratively, the node considers a candidate sleep time t_j, where t_0 has value D'. In each iteration, if t_j is less than one time unit, the iterative procedure ends, and T_{sleep_i} is set to be one time unit. Otherwise, supposing that the sleep time is t_j, the node estimates its distance $\hat{dist}_j$ from the target at the time of the next position update, as follows:

$$\tan\alpha_i = (y_i - y_{i-1})/(x_i - x_{i-1}) \quad (11)$$

$$\hat{\Delta r}_j = Speed_i \cdot t_j \quad (12)$$

$$\hat{\Delta x}_j = \hat{\Delta r}_j \cdot \cos\alpha_i = \hat{\Delta r}_j/\sqrt{1+\tan^2\alpha_i} \quad (13)$$

$$\hat{\Delta y}_j = \hat{\Delta x}_i \cdot \tan\alpha_i \quad (14)$$

$$\hat{y_j} = y_i + \hat{\Delta y}_j \quad (15)$$

$$\hat{x_j} = x_i + \hat{\Delta x}_j \quad (16)$$

$$\hat{dist}_j = \sqrt{(\hat{y_j} - Y)^2 + (\hat{x_j} - X)^2} \quad (17)$$

If $\hat{dist}_j$ is predicted to be a "good" distance for communication (e.g., less than or equal to the historical minimum distance, in the case of the LD algorithm), the iteration ends, and the current t_j is used as the sleep time T_{sleep_i}. Otherwise, we repeat the iterative procedure with t_{j+1} set to t_j minus one time unit.

The experimental results in Section 5 show that by making more elaborate assumptions about a node's movement, PoBA can in fact be less effective than the simpler SBA and DBA approaches. We omit the pseudo-code of PoBA in this paper due to limited space; interested readers may refer to [5] for the details.

4. System Design and Implementation on Sensor Network Platform

4.1. System Design

In prototyping our adaptive scheduler on a sensor device, which is resource constrained, simplicity and efficiency are two important goals. Figure 2 shows the architecture of the software system. The core system component is the *power manager*, which monitors the performance of the movement prediction, receives packets generated by applications, schedules the packet transmission, and puts the system into sleep whenever appropriate. In addition, the power manager powers control the wireless radio interface based on the result of the movement prediction.

The power manager has two main schedulers: a *transmission scheduler* and a *sleep scheduler*. The transmission scheduler schedules pending packets for transmission. The sleep scheduler determines the schedule of position updates; it puts the system to sleep between updates provided that the application is idle. The power manager maintains a *traffic queue* of all the pending packets. In addition, a *traffic table* stores the necessary state information about each packet in the traffic queue (e.g., the packet's remaining delay budget).

Besides the power manager, a *positioning system* component keeps location information about the mobile node for functions of power management and routing. Underlying OS services for the system prototype are provided by TinyOS running on the Berkeley Mica Mote. Implementation details are described in the following section.

4.2. Implementation approach

Our system is implemented on Berkeley mote running TinyOS. Furthermore, we have prototyped a sample application that generates data packets at a constant rate. Each packet has a static deadline. As described, the transmission and sleep schedulers jointly decide when to send the packets generated by the application.

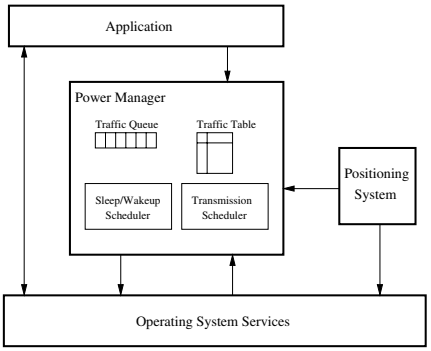


Figure 2. Overview of mobile node system architecture

The prototyped transmission scheduler implements the Least Distance (LD) algorithm, as explained in Section 2. The sleep scheduler puts a sensor node to sleep by using the *Snooze* TinyOS component. It can be programmed to apply either fixed or adaptive sleep periods between position updates.

The current system prototype does not use GPS. Instead, we emulate the movement updates using a software component which periodically generates new position information. The updates are driven by either a Position Generator component running in real-time or by a stored mobility profile. Notice that a system equipped to use GPS will not induce much additional energy overhead as available commercial GPS systems consume as low as $12 \sim 24\ mW$ in full operation [1] (energy consumption is even lower in power saving mode).

Power control of wireless transmission is important. With the Mica Mote, a TR1000 916.5 MHz transceiver is used as the radio interface. With the knowledge of the communication distance from the receiver, the corresponding transmitter output power can be set by a TinyOS component called *Pot*.

5. Performance Evaluation

We evaluate the performance of the proposed adaptive scheduler through both packet-level network simulations and measurement results on our system prototype. For both sets of results, we report the percentage energy saving in communication and the percentage distance saving as the main performance metrics. We also report the average number of wakeups used by the different adaptive sleep schedulers until the pending data are sent. A larger number of wakeups indicates shorter sleep intervals, and hence increased energy consumption. Lastly, energy-efficient communication is achieved by postponing packet transmission. The actual packet delays caused by the postponement strategy are reported for the measurement results.

5.1. Simulation results

Nodal movement in our simulations is defined in a movement file. Unless otherwise specified, all the movement files are generated by the CMU node-movement generation tool *setdest*, using the mobility model given in Section 5.1.1. The mobile node moves in a 150 m by 150 m area. The simulation area is divided into twenty-five 30 m by 30 m grids. A destination node of the network traffic is located in one of the grids. Simulation traffic is generated at continuous bit-rate (CBR) with a packet size of 1500 bytes. Each traffic burst consists of 10 packets with an inter-packet interval time of 0.005 seconds, and the corresponding bit-rate is about 2.4 Mbit/s. A burst is generated every 200 seconds. To calculate the transmission power, the standard *two-ray-ground* model is used as the wireless propagation model with the following parameters: transmit antenna height h_t and receive antenna hight h_r are both 1.5 m; transmit gain G_t and receive gain G_r are both 1.0; frequency f_c is 914.0e6 Hz; λ is 3.0e8/f_c; receiving power threshold is 5.00302e-07 W. Each simulation run lasts 20,000 seconds of simulation time. Five independent runs are repeated for each experiment. The average and confidence interval over the five runs are reported in the results.

5.1.1 Mobility Model

Our experiment uses the random waypoint mobility model with the modifications proposed in [15]. The original random waypoint mobility model fails to maintain a meaningful steady-state average speed of nodal movement. In the modified random waypoint model, lower and upper bounds on the nodal speed, denoted as $speed_{min}$ and $speed_{max}$ respectively, are introduced, and the average nodal speed at steady state can be shown to be about $\frac{speed_{max}-speed_{min}}{2}$.

We further define several mobility scenarios corresponding to plausible nodal speeds in real life. They are specified as walker, runner, bicycle, vehicle(local), vehicle(highway) and airplane. in Table 1.

Table 1. Average Speed of Different Mobility Scenarios

scenario	m/s	km/h	mile/h
walker	2	7.2	4.5
runner	3	10.8	6.8
bicycle	7	25	15.6
vehicle(local)	20	72	45
vehicle(highway)	30	108	67.5
airplane	80	288	180

For each mobility scenario, we set $speed_{min}$ and $speed_{max}$ with four variance settings, namely 10%, 25%, 50% 75% of the average speed. For example, with a 10% variance, the $speed_{min}$ and $speed_{max}$ of the walker scenario are set to be 1.8 m/s and 2.2 m/s, respectively.

5.1.2 Periodic wakeup with fixed sleep periods

We evaluate the performance of the sleep scheduler using fixed sleep periods. We vary the sleep period T_{sleep} and the maximum allowable delay D in a set of experiments, and measure the percentage distance/communication energy savings of the LD algorithm over a vanilla transmission scheduler that sends packets as soon as they arrive. The results for the bicycle scenario are given in Figures 3 and 4.

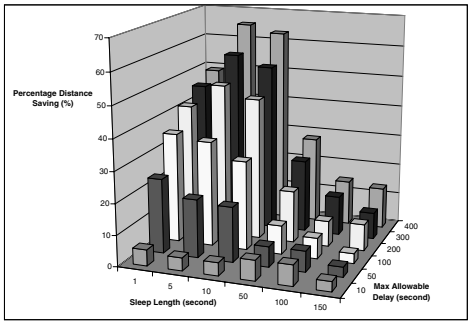


Figure 3. Percentage distance saving as a function of fixed sleep period T_{sleep} and delay budget D.

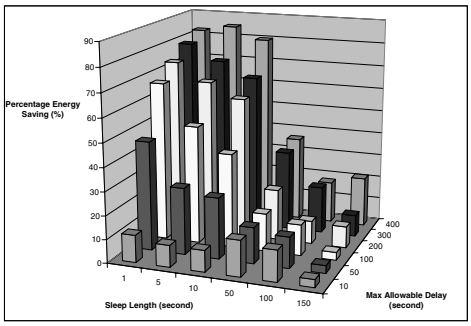


Figure 4. Percentage communication energy saving as a function of fixed sleep period T_{sleep} and delay budget D.

From the figures, the performance of the motion prediction, in terms of both distance and energy savings, generally degrades as the sleep period increases. Particularly, when the sleep period becomes too long, the performance drops dramatically. The results confirm the discussions in Section 3 for fixed periodic sleep.

5.1.3 Performance comparison between sleep strategies

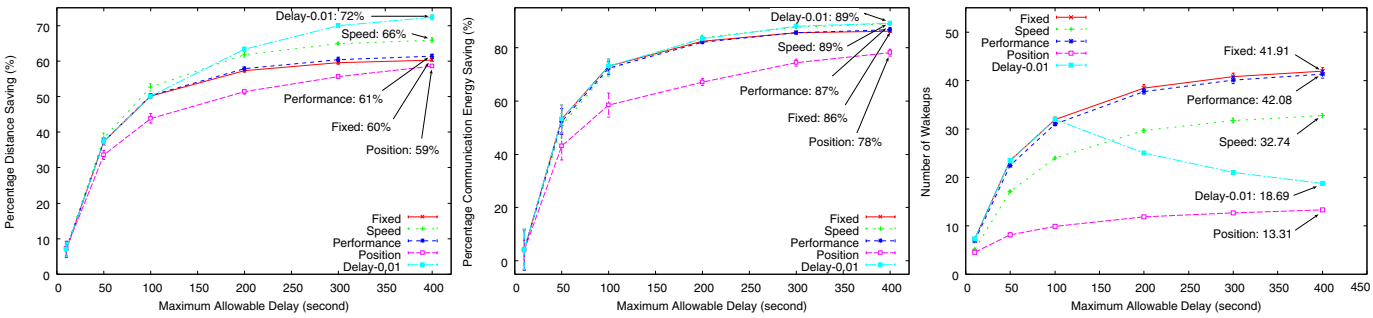
In this section, we compare the performance of fixed periodic sleep and the adaptive sleep algorithms in Section 3. In the experiments, we additionally enforce a minimum sleep time of one second (i.e., if the sleep time calculated by an algorithm is less than one second, we will use one second as the actual sleep time). We report the average energy/distance savings achieved by the LD algorithm, and the number of system wakeups (for position sampling) required by the sleep scheduler.

The reported results are for the bicycle mobility scenario. For fixed periodic sleep, the sleep period of $T_{sleep} = 1$ second is used, since it is shown to achieve the highest communication energy saving given the simulation parameters (see Figure 4). For DBA adaptation, we set $\alpha = 0.01$. We now summarize the results as follows. (In the figures, *Fixed, Speed, Performance, Position, Delay-0.01* denote fixed sleep, SBA adaptation, PeBA adaptation, PoBA adaptation, and DBA adaptation, respectively.)

A. Percentage distance savings. From Figure 5(a), the different sleep algorithms show little difference in distance saving when D is small (less than 10 s). This is because if D is small, the node will have little opportunity to move close to the target before the deadline expires. As D increases, the achieved distance savings increase for the algorithms, albeit at different rates. In general, DBA outperforms the others in most cases, while SBA performs better when D is not too big. For example, when $D = 400$ s, DBA and SBA adaptations outperform fixed periodic sleep by 12% and 6%, respectively, while PoBA and PeBA adaptations has about the same performance as fixed periodic sleep.

B. Percentage communication energy savings. From Figure 5(b), both SBA and DBA adaptations generally perform the best among all the sleep algorithms. The results are consistent with the distance saving results.

C. Number of wakeups. We evaluate the average number of wakeups for each sleep algorithm until the pending data are sent. A larger number of wakeups implies proportionally higher energy consumption by the overall system besides communication. The results are shown in Figure 5(c). Notice that fixed periodic sleep has the highest number of wakeups, since a short sleep period is required to achieve better energy savings in network transmission. Notice also that DBA performs the same as fixed period sleep for D less than about 100 seconds. This is because when D is small, DBA will calculate a small sleep time, and the lower bound sleep time of one second will be used instead,



(a) Percentage distance savings (b) Percentage communication energy savings (c) Number of wakeups

Figure 5. Performance Comparison with different wakeup strategies.

which is the same as the sleep period of the fixed algorithm.

Comparing Figures 5(a) and 5(c), we find that with SBA and DBA adaptations, the system wakes up less frequently (by 20% to 55%) than with fixed periodic sleep. At the same time, both SBA and DBA produce better movement prediction results than fixed sleep (e.g., 5% to 12% higher in percentage distance saving). The results show that by adaptively finding sampling times that are productive, the system in fact outperform sampling times that are strictly periodic, even when it has to work less hard.

In summary, we conclude that, SBA, DBA, and PeBA adaptations generally outperform fixed periodic sleep. Particularly, SBA and DBA adaptations are able to maintain good motion prediction performance (in terms of communication energy/distance savings) with fewer wakeups.

5.1.4 Performance comparison between mobility scenarios

The previous results are taken for the bicycle mobility scenario. We now evaluate the performance of the sleep schedulers under the different mobility scenarios presented in Section 5.1.1, namely walking person, runner, bicycle, vehicle (local), vehicle (highway), and airplane. The maximum allowable delay D in the simulations is 200 seconds, and the nodal speed variance is 25%.

Figure 6(a) illustrates the percentage distance saving for the different mobility scenarios. The saving generally increases as the mobility scenario changes from low to high speed. This is expected, since by moving faster, a node may have more opportunities to move to a better sending position before the deadline expires. We also notice that as the nodal speed increases, fixed periodic sleep, SBA, and PeBA achieve the best distance savings. This is because a higher nodal speed can cause significant position changes to happen more frequently, and hence can benefit from more frequent position updates. Fixed periodic sleep is able to check frequently in this experiment because a small sleep period is used, as discussed above. As regards SBA and PeBA, both algorithms are designed explicitly to use a higher sampling frequency as the nodal speed increases.

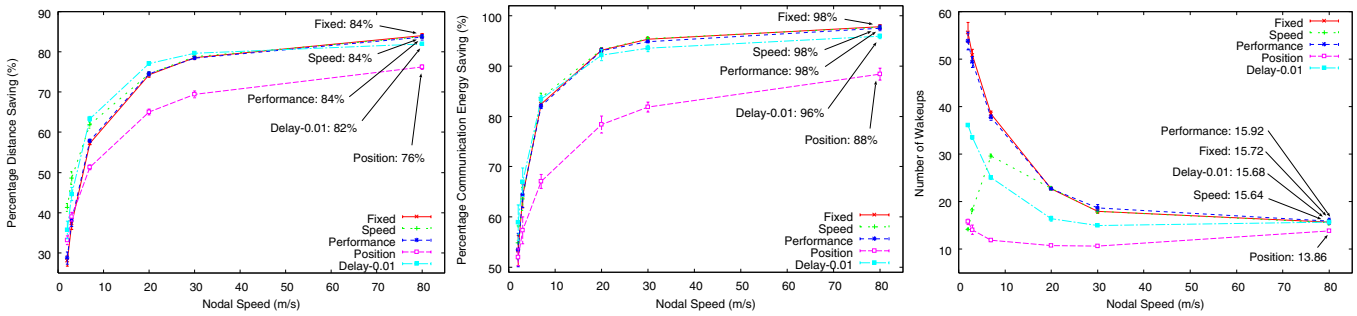
The percentage communication energy savings in Figure 6(b) show similar performance as the percentage distance saving in Figure 6(a). This follows from the observation that a closer communication distance will likely result in lower energy use.

Figure 6(c) shows the number of wakeups for the different mobility scenarios. In general, the number of wakeups decreases as the nodal speed increases. This is because as a node moves faster, it will generally take less time to move close to the target for communication. Therefore, there are fewer wakeups and a smaller postponement delay before the communication is carried out. From Figure 6(c), notice also that unlike the other sleep algorithms, SBA adaptation wakes up more times as the nodal speed increases over the low to moderate speed range. When the nodal speed further increases, however, the number of SBA wakeups begins to decrease. This is because the lower bound sleep period of one second will be used at the high nodal speed.

In summary, under low speed mobility scenarios, SBA, DBA, and PeBA adaptations outperform fixed periodic sleep in terms of higher distance and communication energy savings and fewer wakeups. Under high speed mobility scenarios, SBA, DBA, and PeBA, adaptations perform as well as fixed periodic sleep, but require less frequent wakeups. Overall, the adaptive schedulers outperform fixed periodic sleep under all the mobility scenarios studied.

5.1.5 Summary of simulation results

To summarize the simulation results, we notice that SBA and DBA adaptations significantly outperform fixed periodic sleep in terms of better savings in communication energy and distance, while requiring less sampling effort. The improved performance appears robust in that it is evident in all of the movement scenarios studied in this paper and



(a) Percentage distance saving with different wakeup strategies.

(b) Percentage energy saving with different wakeup strategies.

(c) Number of wakeups with different wakeup algorithms.

Figure 6. Performance Comparison between mobility scenarios

the extended results in the technical report [5]. In general, PoBA does not perform as well in saving communication distance and energy, because it can make too specific assumptions about the future movement, which may not be exactly true in practice.

5.2. Measurement results on sensors

5.2.1 Experimental setup

We measure the energy savings of the prototype mote system described in Section 4 The system energy consumption is measured as follows. A constant power supply provides 3.0 Volts of power to the mote. Between the power supply and the mote, we connect in series a small resistor of 0.05 Ohm to the power input line of the mote. The voltage across the resistor is measured by an HP multimeter. The current running through the power input line can then be calculated as: $I = U_{measure}/R$. Since the power input has a constant voltage of U_{input}, the power can be calculated as $P = U_{input} \cdot I$, while the corresponding energy consumption is $E = U_{input} \cdot I \cdot T_{measure}$.

In our experimental setup, packets are generated at a source mote and sent to a receiver mote serving as the target. The target forwards the received packets to a connecting PC for further processing and analysis. The source sensor node does not physically move in our experiments. Instead, we emulate movement using a *positioning* software component which updates the position of the mobile node in real time through a stored mobility profile. The vehicle(local) mobility scenario is used to drive an experiment. The sender then regards itself as moving according to the position updates, and when sending, powers control the packet transmissions according to the emulated positions.

Application packets are generated at the source node as follows. Every 16 time units, where each time unit is 1/16 second, a burst of ten 36-byte packets arrive back to back. This corresponds to an average data rate of 10 packets/s. All the packets in a burst receive the same static deadline of $D = 32$ time units after arrival. Each experiment runs for 3,200 time units for a total of 200 bursts or 2000 packets. Measurement results are taken as averages over 200 packet bursts. In an experiment, we record all the multimeter readings of the voltage across the small resistor, and their average is calculated. Each experiment is repeated three times, and the reported power numbers are averages over the three runs.

5.2.2 Performance comparison

Since DBA is found to perform well in simulation, we now compare the performance of DBA with fixed periodic sleep on the system prototype. For the fixed strategy, the sleep period is varied to be 1, 2, 4, and 8 time units. The performance metrics are the percentage distance saving, the number of sampling wakeups until the pending data are sent, the actual postponement delay, and the power of the actual system measured. A baseline system without applying sleep scheduling is also measured for comparison.

Table 2. Comparison of Adaptive Wakeup scheduler, Fixed Wakeup scheduler and Non-sleep

Scheduler	Power (mW)	Distance Saving	# of Wakeup	Delay
Baseline	18.042	85.2%	0	4.83
Fixed ($T = 1$)	9.642	85.2%	5.71	4.83
Fixed ($T = 2$)	9.633	53.3%	4.67	7.34
Fixed ($T = 4$)	8.874	40.7%	3.54	10.16
Fixed ($T = 8$)	8.904	5.1%	1.8	6.40
DBA	8.193	44.8%	3.17	9.66

The measurement results are shown in Table 2. The results show that both the fixed and DBA sleep algorithms significantly reduce (by 45%–55%) the energy consumption of

the total system compared with the baseline system. Notice also that as T increases from 1 to 4 time units, the total energy use decreases, as more energy is saved by sleeping. As T increases from 4 to 8 time units, however, the energy use instead *increases* slightly. This is because the energy saved by sleeping is now outweighed by the increased energy of *communication* as movement prediction becomes less effective with a longer sleep. Hence, total system energy use increases. The situation shows the need and difficulty of determining an optimal sleep period using the fixed strategy. The DBA algorithm, however, is able to adapt its sleep schedule to minimize wakeup and yet keep the effectiveness of the LD postponement algorithm. Hence, the measured energy use of the *total* system is reduced significantly, by 8% to 15% compared with the best fixed periodic sleep algorithm, and by 55% compared with the baseline system.

6. Related Work

The problem of limited battery on mobile sensor devices has recently received a lot of attention. Much research, in particular, has focused on optimizing networking protocols for energy efficiency (e.g., [8]).

Previous work on mobility prediction [2, 9–11] has focused on resource reservation and quick handoff management between base stations to provide QoS support for mobile wireless users. In [12], motion prediction is used to minimize disruptions induced by frequent changes in ad hoc network topology, and to rapidly and proactively reconstruct routes based on predictions of the future network topology. In [7], mobility is exploited to increase the capacity of ad hoc wireless networks. Their work addresses QoS and reduced routing overhead. They do not address the issue of energy conservation for mobile sensor devices. Wakeup schedules have been studied by [4,13,14,16]. They are concerned with improving network connectivity, end-to-end delay, and network throughput, and therefore have a different focus than our work.

7. Conclusion

Effective and energy-efficient sampling of node movement enables motion-predicted communication, which can save significant energy of wireless transmission. In this paper, we have presented an adaptive scheduler for determining an effective sampling schedule given dynamic system conditions. Between two sampling instants, a mobile node can be put to sleep, conserving power. Our implementation and simulation results show that the adaptive scheduler can achieve substantial energy savings over a system that does not exploit opportunities to sleep. Moreover, the scheduler's adaptivity allows it to outperform fixed sleep periods between sampling, since the "right" period depends on dynamic system conditions and cannot be determined *a priori*. The motion tracking enabled by our sleep scheduler has applications beyond energy management. For example, it can enable the prediction of location-dependent wireless channel conditions (e.g., interference) to improve transmission throughput.

References

[1] *http://rfdesign.com/news/integrated-GPS-receiver/*

[2] A. Bhattacharya and S. K. Das, "LeZi-Update: An Information-theoretic approach to track mobile users in PCS networks," in *Proc. ACM/IEEE MobiCom*, Aug. 1999, pp. 1-12

[3] S. Chakraborty, Y. Dong, D. K. Y. Yau, and J. C. S. Lui, "On the Effectiveness of Movement Prediction To Reduce Energy Consumption in Wireless Communication," *IEEE Transactions on Mobile Computing*, To Appear.

[4] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris, "Span: An Energy-Efficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Networks," in *Proc. ACM/IEEE MobiCom*, 2001

[5] Y. Dong and D. K. Y. Yau, "An adaptive sleep scheduler for energy-efficient movement-predicted wireless communication", Technical Report, CS Dept, Purdue University, West Lafayette, IN, April 2005.

[6] P. R. Freeman, "The secretary problem and it's extensions: A review," *International Statistical review 51, 189-206, 1983*

[7] M. Grossglauser and D. Tse, "Mobility Increases the Capacity of Ad-hoc Wireless Networks," in *IEEE/ACM Transactions on Networking*, vol. 10, no. 4, August, 2002, pp. 477-486.

[8] D. B. Johnson, D. A. Maltz and J. Broch, "DSR: The Dynamic Source Routing Protocol for Multi-Hop Wireless Ad Hoc Networks," *Ad Hoc Networks*, edited by C. E. Perkins, Chap. 5, pp. 194-172, Addison-Wesley, 2001

[9] D. Levine, I. Akyildiz and M. Naghshineh, "A Resource Estimation and Call Admission Algorithm for Wireless Multimedia Networks Using the Shadow Cluster Concept," *IEEE/ACM Trans. on Networking*, 5(1), Feb. 1997, pp 1-12.

[10] G. Liu and G. Maguire Jr., "A Class of Mobile Motion Prediction Algorithms for Wireless Mobile Computing and Communications," *ACM/Baltzer MONET*, 1(2), 1996, pp. 113-121

[11] T. Liu, P. Bahl and I. Chlamtac, "Mobility Modeling, Location Tracking, and Trajectory Perdition in Wireless ATM Networks," *IEEE JAC*, 16(6), Aug. 1998, pp. 922-936

[12] W. Su, S. J. Lee and M. Gerla, "Mobility Prediction and Routing in Ad Hoc Wireless Networks", *International Journal of Network Management*, Wiley & Sons, 2000

[13] X. Yang, and N. Vaidya, "A Wakeup Scheme for Sensor Networks: Achieving Balance between Energy Saving and End-to-end Delay," in *Proc. the 10th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS 2004)*, pp. 19-26, May 2004

[14] W. Ye, J. . Heidemann, and D. Estrin, "An Energy-Efficient MAC Protocol for Wireless Sensor Networks," in *Proc. IEEE Inforcom*, 2002

[15] J. Yoon, M. Liu and B. Noble, "Random Waypoint Considered Harmful," in *Proc. IEEE INFOCOM*, Apr. 2003

[16] R. Zheng, J. C. Hou, and L. Sha, "Asynchronous Wakeup for Ad Hoc Networks: Theory and Protocol Design," in *Proc. MobiHoc'03*, 2003