SLEEPING POLICIES FOR ENERGY-EFFICIENT TRACKING IN SENSOR NETWORKS

Arun Visvanathan and Venugopal V. Veeravalli

University of Illinois at Urbana-Champaign
ECE Department and Coordinated Science Laboratory
1308 West Main Street
Urbana, Illinois, 61801-2307

ABSTRACT

The problem considered is that of tracking an object that is moving through a dense network of wireless sensors. Each sensor has a limited range for detecting the presence of the object and the goal is to track the location of the object to within the accuracy of the range of the sensor. The network is assumed to be sufficiently dense so that the sensors cover the entire area of interest. The object is assumed to follow a random path through the sensor field whose statistics are assumed to be known *a priori*. It is assumed that in order to conserve energy the sensors go into sleep mode periodically. The sleep duration is determined at the time the sensor goes to sleep based on the information available to the sensor. It is assumed that a sensor that is asleep cannot be communicated with or woken up prematurely. Having sleeping sensors in the network could result in tracking errors, and hence there is a natural tradeoff between energy savings and tracking error. This paper considers the design of sleeping policies with the goal of optimizing this tradeoff.

Keywords : distributed signal processing, wireless sensors, sleep mode, MMSE tracking.

1. INTRODUCTION

Advances in technology are enabling the deployment of vast sensor networks through the mass production of cheap sensor units with small batteries. Such sensor networks can be used in a variety of application areas, including medicine, search and rescue, environmental protection, military, manufacturing, home security, gaming and entertainment. Our focus in this paper is on applications of sensor networks that involve *tracking*, e.g., surveillance, wildlife studies, environmental control, and health care. For example, sensor networks can be deployed in forests and oceans to study and monitor various wildlife species, without actively intervening in their daily lives. Such studies can be a valuable source of knowledge for scientists and biologists.

The sensors get their energy through small embedded batteries or from the environment (solar, vibrations, etc.). Regardless of their power source, the sensor nodes typically need to operate on limited energy budgets. In order to conserve energy, the sensors may be put into sleep mode whenever they are not required to be active. If the sensors were always awake we could of course achieve perfect tracking (with zero tracking error) since the sensors completely cover the field. The problem of interest is to trade off tracking error with energy conservation via sleeping.

One way to effect the transition between the active and sleep modes is to have the sensor be in sleep mode by default and to use a low power radio channel to wake up (page) the sensor when it is needed. To the best of our knowledge, this approach has been taken in all of the literature to date on energy-efficient tracking using sensor networks. In [1] and [2], the authors have considered a dynamic convoy tree based approach. The tree consists of the sensors that are awake, and it is updated periodically based on the position and velocity of the object. A cell based distributed algorithm is developed in [3], where the sensors in neighboring cells to the one in which the object is present are woken up based on the trajectory of the object. A model for sensing cost for sensors is assumed in [4], and a collaborative tracking algorithm that minimizes the sensing cost is developed to choose the sensor that will be awake and track the object in the next time instant, based on the position and velocity of the object. In [5], the authors propose a method for the estimation of position using a triangulation method, and the estimation of velocity via position information for more than one time instant. This estimated data is then used to warn other sensors about the motion of the object so that they can be alert and take appropriate actions. Tracking objects by the method of localized prediction, with a localized sensor network architecture ,where most of the sensor nodes keep sleeping until woken up by an active sensor node is proposed by the authors in [6]. In [7], the sensor network is divided into clusters, and a protocol that uses a predictive mechanism is devised to intimate cluster heads about approaching targets. Based on the prediction of trajectories of the target, the cluster head activates the most appropriate sensors before the arrival of the target.

This research was supported by the National Science Foundation under the award CCF-0049089, through the University of Illinois.

The design problem that we consider is very different from the one considered in the prior work described above for the following reason. We assume that using a wake-up channel to effect the transition between the active and sleep modes is *impractical* given current sensor technology. This is because the power consumed by the wake-up channel is generally not negligible compared to the power consumed by the active sensor [8]. Alternatives to the wake-up channel approach are: (i) have the sensor enter and exit the sleep mode using a fixed or a random duty cycle, and (ii) use prior knowledge about the sensor observations to determine the sleeping strategy. The latter approach is clearly more effective than the second if prior knowledge about the observations is available, and this is the approach we take in this paper.

2. PROBLEM DESCRIPTION

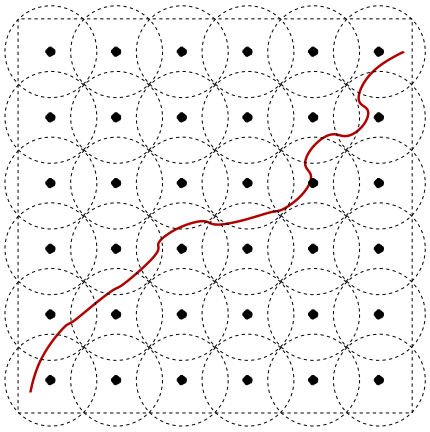

Fig. 1. Object tracking in a field of sensors.

The tracking problem of interest is one where the sensors are distributed in a two-dimensional plane. Each sensor has a limited range for detecting the presence of the object being tracked, and the goal is to track the location of the object to within the accuracy of the range of the sensor. For such a tracking strategy to be well-posed we need to assume that the sensor field is sufficiently dense so that the sensors cover the entire area of interest (see Figure 1). The object follows a random path through the sensor field whose statistics are assumed to be either known *a priori* or estimated on-line. The goal is to design the sleeping policies at the sensors to optimize the tradeoff between energy consumption and tracking error. The design problem is more easily introduced for tracking in one dimension. We therefore first consider this simpler one-dimensional case, and then generalize the resulting policies to two dimensions.

Consider a one-dimensional sensor network with sensors placed unit distance apart from $-b$ to $+b$. An object that has to be tracked by this sensor network is assumed to undergo a symmetric random walk; i.e, at each time instant, the object moves a unit distance either to the left or to the right with equal probability. A central unit[1], which controls this sensor network, is assumed to maintain the information required to compute the sleep times of the sensors in the system and to assign the sleep times for the sensors that come awake. A sensor is either awake or sleeping at each time instant. If the object is within the range of a sensor that is awake, the sensor detects the object and sends this information to the central unit. Initially it is assumed that the object is at position zero and that all the sensors are awake.

One simple policy to track the object is the "always track" policy where the sleep time of a sensor that has come awake is set as the distance between the sensor and the object, when the object is not present at that sensor's location. This is because the object will be able to move at most n units of distance away from its present position during n time units for any given positive integer n. If the object is present at a sensor's location then the sleep time of that sensor is set to two, because the object has to move to one of the adjacent positions before it comes back to the same position. If we set the sleep time to be larger than the above specified time, then the network might not be able to track the object at all times, leading to tracking error. But sleeping for more time will decrease the energy spent per unit time in the network. Thus there is a tradeoff between energy consumption and tracking error for this system.

3. STATE SPACE DESCRIPTION

Let $\phi_{k,l}$ and $u_{k,l}$ denote the residual sleep time and the input sleep time, respectively, of the sensor located at position $l \in \{-b, -b+1, \ldots, b-1, b\}$ at time instant k. If a sensor has non-zero residual sleep time, then it sleeps during the current time instant. Also, its residual sleep time at the next time instant gets decremented by one. If the residual sleep time is zero, then the sensor is awake at the current time instant. Also the sleep time at the next time instant is set to the input sleep time. Therefore, the evolution of $\phi_{k,l}$ is given by

$$\phi_{k+1,l} = (\phi_{k,l} - 1)\ \mathbf{1}_{\{\phi_{k,l} \neq 0\}} + u_{k,l}\ \mathbf{1}_{\{\phi_{k,l}=0\}} \quad (1)$$

where $\mathbf{1}$ is the indicator function. The position of the object m_k evolves as

$$m_{k+1} = m_k + w_k \quad (2)$$

where the w_k's are i.i.d Bernoulli random variables taking values $+1$ and -1 with equal probability for a symmetric random walk. To simplify the notation, let

$$\Phi_k = [\phi_{k,-b}, \phi_{k,-b+1}, \ldots, \phi_{k,b-1}, \phi_{k,b}]^T$$

[1]Generalizations to the distributed scenario without a central control unit are discussed in Section 9

and

$$U_k = [u_{k,-b}, u_{k,-b+1}, \ldots, u_{k,b-1}, u_{k,b}]^T.$$

The state of the system is then given by $X_k = (\Phi_k, m_k)$. Also, we assume that the problem terminates if the object exits the system. Therefore, the state update from (1) and (2) takes the form

$$X_{k+1} = \begin{cases} f(X_k, U_k, w_k) & \text{if } X_k \neq t \text{ and } U_k \neq \textit{terminate}, \\ t & \text{if } X_k = t \text{ or } U_k = \textit{terminate}. \end{cases} \tag{3}$$

At any time instant, we assume that the object's position can be observed only when the sensor present at the object's position is awake during that instant (or when the object exits the system). The observation at time instant k can be represented as $Z_k = (\Phi_k, s_k)$, where s_k is equal to either the position of the object when it is known or an erasure ϵ, when the object's position is unknown. Thus,

$$\begin{aligned} s_k &= m_k \, \mathbf{1}_{\{m_k = -b-1 \, or \, b+1 \text{ or } \phi_{k,m_k}=0\}} \\ &\quad + \epsilon \, \mathbf{1}_{\{m_k \neq -b-1 \, and \, b+1 \text{ and } \phi_{k,m_k} \neq 0\}} \end{aligned} \tag{4}$$

At any given time instant, there are two types of costs that are incurred - the energy cost and the tracking cost. We assume that an energy cost of unity is contributed by each sensor that is awake, and a tracking cost of c is incurred when the object is not tracked. Different values of c give different points on the trade-off curve between the average energy spent and the average tracking error. The total cost incurred at time instant k is given by

$$g(X_k, U_k, w_k) = \mathbf{1}_{\{X_k \neq t\}} \left[c \, \mathbf{1}_{\{\phi_{k,m_k} \neq 0\}} + \sum_{l=-b}^{b} \mathbf{1}_{\{\phi_{k,l}=0\}} \right] \tag{5}$$

The information available during the current time instant to choose the optimal sleeping policy consists of present and past observations, and the past input sleep times. Thus the current sleep time can be chosen as

$$U_k = \mu_k(I_k) \tag{6}$$

where

$$\begin{aligned} I_k &= [Z_0, Z_1, \ldots, Z_k, U_0, U_1, \ldots, U_{k-1}]^T, & (7) \\ I_0 &= Z_0. & (8) \end{aligned}$$

We assume an infinite horizon discounted cost formulation of the optimization problem with forgetting factor $\alpha \in (0, 1)$. The cost function to be optimized is given by

$$J_0(I_0, \mu_0, \mu_1, \ldots) = \mathop{\mathrm{E}}_{|I_0} \left[\sum_{k=0}^{\infty} \alpha^k g\left(X_k, \mu_k(I_k), w_k\right) \right] \tag{9}$$

where $\mathop{\mathrm{E}}_{|I_0}$ represents the expectation over the distribution of $(X_0, X_1, \ldots, w_0, w_1, \ldots)$, conditioned on I_0. The fraction $\frac{1}{1-\alpha}$ denotes the approximate time period over which the chosen sleeping policy is optimized. Hence α is chosen such that the above fraction is equal to the expected time that the object is present in the system. The optimal cost is given by

$$J_0^*(I_0) = \min_{\mu_0, \mu_1, \ldots} J_0(I_0, \mu_0, \mu_1, \ldots) \tag{10}$$

The input sleep times will not affect the sensors that are asleep. Therefore we can reduce the input space by searching for optimal sleeping policy only over those set of input sleep times which are zero for sensors that are asleep. Therefore it is enough to minimize the cost function $J_0(I_0, \mu_0, \mu_1, \ldots)$ over the set of sleeping policies $U_k \in U(I_k)$, where

$$U(I_k) = \begin{cases} \emptyset & \text{if } X_k = t, \\ \{\textit{terminate}, V(I_k)\} & \text{if } X_k \neq t. \end{cases} \tag{11}$$

$$\begin{aligned} V(I_k) &= \{\Gamma_k = [\gamma_{k,-b}, \gamma_{k,-b+1}, \ldots, \gamma_{k,b-1}, \gamma_{k,b}]^T \\ &\qquad \Big| \, \forall l, \, \gamma_{k,l} = 0 \text{ if } \phi_{k,l} \neq 0\} \end{aligned} \tag{12}$$

From the dynamic programming argument, the optimal cost function can be recursively computed as

$$\begin{aligned} J_k^*(I_k) = \min_{U_k \in U(I_k)} \Big\{ & t(I_k), \\ & \mathop{\mathrm{E}}_{|I_k} \left[g(X_k, U_k, w_k) + \alpha \, J_{k+1}^*(I_k, Z_{k+1}, U_k) \right] \Big\} \end{aligned} \tag{13}$$

where

$$t(I_k) = \begin{cases} \frac{2b+1+c}{1-\alpha} & \text{if } s_k \neq -b-1 \text{ and } b+1 \\ 0 & \text{if } s_k = -b-1 \text{ or } b+1 \end{cases} \tag{14}$$

The justification for using $\frac{2b+1+c}{1-\alpha}$ in (14) comes from the fact that the optimal cost-to-go function $J_k^*(I_k) < \frac{2b+1+c}{1-\alpha}$, $\forall k, I_k$. Therefore we will terminate only when the object goes out of the system.

4. SUFFICIENT STATISTICS

We cannot use I_k directly because it grows indefinitely with time. Thus we have to find a suitable transformation of I_k that can be used instead of I_k, to compute the optimal policy without loss in performance. Let

$$\begin{aligned} p_{k,l} &= \Pr\{\text{object is at position } l \text{ at time } k | I_k\} & (15) \\ q_{k,l} &= \Pr\{\text{object is at position } l \text{ at time } k{+}1 | I_k\} & (16) \\ P_k &= [p_{k,-b-1}, p_{k,-b}, \ldots, p_{k,b}, p_{k,b+1}]^T & (17) \\ Q_k &= [q_{k,-b-1}, q_{k,-b}, \ldots, q_{k,b}, q_{k,b+1}]^T & (18) \end{aligned}$$

The probabilities P_k and Q_k can be shown to evolve as

$$q_{k,l} = \begin{cases} \frac{1}{2}[p_{k,l-1} + p_{k,l+1}] & \text{if } l \neq (-b-1), (b+1) \\ \frac{1}{2}p_{k,-b} & \text{if } l = -b-1 \\ \frac{1}{2}p_{k,b} & \text{if } l = b+1. \end{cases}$$

$$p_{k,l} = \begin{cases} \mathbf{1}_{\{s_k=l\}} + \mathbf{1}_{\{s_k=\epsilon\}} \mathbf{1}_{\{\phi_{k,l}\neq 0\}} \frac{q_{k-1,l}}{\sum_m \mathbf{1}_{\{\phi_{k,m}\neq 0\}} q_{k-1,m}} \\ \qquad \text{if } l \neq -b-1 \text{ and } b+1, \\ \mathbf{1}_{\{s_k=l\}} \qquad \text{if } l = -b-1 \text{ or } b+1. \end{cases}$$

It can be shown that $\Psi_k = (\Phi_k, P_k)$ is a sufficient statistic for the dynamic programming problem.

Remark 4.1. *We can now transform the problem with partially observable state-space to a problem with completely observable state-space by considering Ψ_k as the state of the system and s_{k+1} as the conditionally independent random noise generated at time k.*

Let ω_k be the fictitious random variable which has the same probability distribution as s_{k+1} given the current state Ψ_k. Then,

$$\Psi_{k+1} = \begin{cases} \tilde{f}(\Psi_k, U_k, \omega_k) & \text{if } \Psi_k \neq t \text{ and } U_k \neq \textit{terminate}, \\ t & \text{if } \Psi_k = t \text{ or } U_k = \textit{terminate}. \end{cases} \tag{19}$$

$$U_k = \tilde{\mu}_k(\Psi_k) \tag{20}$$

$$\tilde{J}^*_k(\Psi_k) = \min_{U_k \in \tilde{U}(\Psi_k)} \Big\{ \tilde{t}(\Psi_k), \mathop{\mathrm{E}}_{\omega_k|\Psi_k} \Big[\tilde{g}(\Psi_k, U_k, \omega_k) + \alpha\, \tilde{J}^*_{k+1}(\tilde{f}(\Psi_k, U_k, \omega_k)) \Big] \Big\} \tag{21}$$

$$\tilde{t}(\Psi_k) = \begin{cases} \frac{2b+1+c}{1-\alpha} & \text{if } p_{k,-b-1} \neq 1 \text{ and } p_{k,b+1} \neq 1 \\ 0 & \text{if } p_{k,-b-1} = 1 \text{ or } p_{k,b+1} = 1 \end{cases} \tag{22}$$

$$\tilde{U}(\Psi_k) = \begin{cases} \emptyset & \text{if } \Psi_k = t, \\ \{\textit{terminate}, \tilde{V}(\Psi_k)\} & \text{if } \Psi_k \neq t. \end{cases} \tag{23}$$

$$\tilde{V}(\Psi_k) = \big\{ \Gamma_k = [\gamma_{k,-b}, \gamma_{k,-b+1}, \ldots, \gamma_{k,b-1}, \gamma_{k,b}]^T \;\big|\; \forall l, \gamma_{k,l} = 0 \text{ if } \phi_{k,l} \neq 0 \big\} \tag{24}$$

$$\tilde{g}(\Psi_k, U_k, \omega_k) = \sum_{l=-b}^{b} \mathbf{1}_{\{\phi_{k,l}=0\}} + c\, \big[1 - \mathbf{1}_{\{\exists l,\, p_{k,l}=1 \text{ and } \phi_{k,l}=0\}}\big] \tag{25}$$

$$\Pr[\omega_k = l \,|\Psi_k] = q_{k,l}\, \mathbf{1}_{\{\phi_{k,l}=1 \text{ or } (\phi_{k,l}=0 \,and\, u_{k,l}=0)\}} \tag{26}$$

$$\Pr[\omega_k = \epsilon \,|\Psi_k] = \sum_{l=-b-1}^{b+1} q_{k,l} \cdot \mathbf{1}_{\{\phi_{k,l}>1 \text{ and } (\phi_{k,l}=0 \,and\, u_{k,l}\neq 0)\}} \tag{27}$$

5. SUCCESSIVE APPROXIMATION

It can be shown that the optimal cost-to-go function does not depend upon the time index given the current state, i.e., for all possible states of the system λ, we have $\tilde{J}^*_k(\lambda) = \tilde{J}^*_{k+1}(\lambda)$, $\forall k$. Therefore, We can further simplify equation (21) to obtain the Bellman equation.

$$\tilde{J}^*(\Psi) = \min_{U \in \tilde{U}(\Psi)} \Big\{ \tilde{t}(\Psi), \mathop{\mathrm{E}}_{\omega|\Psi} \Big[\tilde{g}(\Psi, U, \omega) + \alpha\, \tilde{J}^* \left(\tilde{f}(\Psi, U, \omega) \right) \Big] \Big\} \tag{28}$$

Also one can show that the mapping $T(W)(\Psi) = \min_{U \in \tilde{U}(\Psi)} \left\{ \tilde{t}(\Psi), \mathop{\mathrm{E}}_{\omega|\Psi} \Big[\tilde{g}(\Psi, U, \omega) + \alpha\, W \left(\tilde{f}(\Psi, U, \omega) \right) \Big] \right\}$ is a contraction mapping. Therefore, the optimal cost and sleeping policy can be obtained by starting with $W_0(\Psi) = 0$, $\forall \Psi$ and recursively applying $W_{n+1} = T(W_n)$, till it converges.

For a problem with seven sensors with maximum sleep time of 10 and probability mass function quantized to multiples of 0.1, there are about 10^9 possible states. Therefore, it is infeasible to compute the optimal sleeping policy even for a small problem with seven sensors. The number of possible sleep times grows exponentially with the number of sensors, and each sleep time is associated with some number of probability mass functions, resulting in this huge state space. In the next section, we have develop a locally-optimal policy that does not use the sleep times of all the sensors.

6. LOCALLY-OPTIMAL SLEEPING POLICY

Let $\binom{a}{b}$ for a non-negative integer a, and a non-negative real number b be defined as

$$\binom{a}{b} = \begin{cases} \frac{a!}{b!\,(a-b)!} & \text{if b is an integer and } b \leq a, \\ 0 & \text{otherwise.} \end{cases} \tag{29}$$

Let the object be at position r at time instant k. To get to position l at time instant $(j+k)$ for $j \geq |l-r|$, j and $|l-r|$ must be both even or odd. The object has to take $\frac{j+|l-r|}{2}$ steps in the direction of position l from position r and $\frac{j-|l-r|}{2}$ in the opposite direction.

$$\Pr\left\{m_{(j+k)} = l | m_k = r\right\} = \binom{j}{\frac{j-|l-r|}{2}} \left(\frac{1}{2}\right)^j \tag{30}$$

Instead of the actual position of the object, if we know the probability mass function of the object's position P_k at time k, then

$$\Pr\left\{m_{(j+k)} = l | P_k\right\} = \sum_{r=-b-1}^{b+1} p_{k,r} \binom{j}{\frac{j-|l-r|}{2}} \left(\frac{1}{2}\right)^j \tag{31}$$

Therefore, the expected total tracking cost F incurred by the sensor at position l by choosing the input sleep time $u_{k,l}$ is given by

$$F = c \sum_{r=-b-1}^{b+1} \sum_{j=|l-r|+\mathbf{1}_{\{l=r\}}}^{u_{k,l}} p_{k,r}\left(\frac{j}{\frac{j-|l-r|}{2}}\right)\left(\frac{1}{2}\right)^j \tag{32}$$

The average expected tracking cost incurred by the sensor during this period is $\frac{F}{u_{k,l}+1}$. The average energy cost incurred during this period is $\frac{1}{u_{k,l}+1}$. The total average expected cost $\hat{J}_l(P_k, u_{k,l})$ contributed by sensor at position l by choosing the input sleep time $u_{k,l}$ is given by

$$\hat{J}_l(P_k, u_{k,l}) = \frac{c \sum_{r=-b-1}^{b+1} \sum_{j=|l-r|+\mathbf{1}_{\{l=r\}}}^{u_{k,l}} p_{k,r}\left(\frac{j}{\frac{j-|l-r|}{2}}\right)\left(\frac{1}{2}\right)^j + 1}{u_{k,l}+1} \tag{33}$$

The input sleep time can be chosen as

$$u_{k,l} = \underset{u}{\operatorname{argmin}}\, \hat{J}_l(P_k, u) \tag{34}$$

We refer to the policy resulting from this minimization as the *locally-optimal* policy.

In computing the locally-optimal policy we note that the expected number of tracking errors as a function of sleep time u first increases, reaches a maximum and decreases back to zero. The reason is that as $u \to \infty$, the average number of times the object will be present at a particular sensor's location tends to zero. On the other hand the energy cost is a decreasing function of the sleep time. Therefore, the local cost function $\hat{J}_l(P_k, u)$ is not convex and has more than one minimum. After the first minimum the cost function increases and then decreases back to zero as $u \to \infty$. Choosing $u = \infty$ is a greedy policy that optimizes the cost for a given particular sensor assuming that the object will stay within the system for an infinite time duration. Therefore choosing the value of u corresponding to the first minimum gives a meaningful locally-optimal sleeping policy.

7. MINIMUM SLEEP TIME POLICY

The locally-optimal sleeping policy described in the previous section is computationally more tractable than the optimal sleeping policy, and can be computed and simulated for reasonably large values of b as we see in Section 8. However, even this simpler policy becomes computationally intractable for large values of b, and would be difficult to compute in real time for implementation in a system. The intractability is further accentuated for the more complex scenarios of tracking an object that moves with a variable velocity in one dimension, and for tracking an object in a two-dimensional environment. We therefore develop a further simplification of the locally-optimal which is computationally tractable and practical.

If we know the position of the object, the locally-optimal sleep time for a sensor that has come awake can be obtained easily as simply a function of the distance of the sensor to the current position of the object. What we now need is a simple heuristic to replace the location of the object when we do not know the exact position of the object. The heuristic that we use is the location that results in the minimum sleep time among the set of possible locations where the object can be located. We call this policy the *minimum sleep time* policy. We note that this minimum sleep time policy can easily be generalized to more complicated tracking scenarios, including the variable velocity and two-dimensional scenarios.

8. SIMULATION RESULTS

We first considered the a one-dimensional sensor network with $b = 20$ with the object undergoing a symmetric random walk. We simulated the the locally-optimal and the minimum sleep time policies and obtained the tradeoff curves for energy versus tracking costs (see Figure 2).

We then considered the more complex scenario of tracking an object undergoing random walk with $w_k = \{-1, 0, 1\}$ with equal probability by following the steps in Section 6. While the locally-optimal policy can be derived in this case, we found that simulating it was computationally infeasible. We therefore developed the minimum distance policy (given in Section 7), simulated it and plotted the tradeoff curve (see Figure 3).

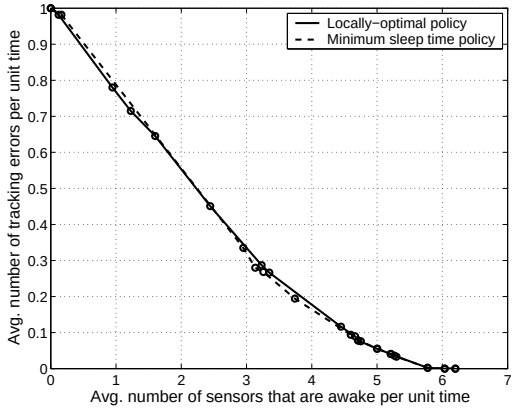

Fig. 2. Trade-off curve for a 1-D sensor network (b=20) tracking an object undergoing a symmetric random walk.

The following observations can be made from the tradeoff curves:

When c is small compared to the average number of sensors that have to be awake for the "always track" policy, the

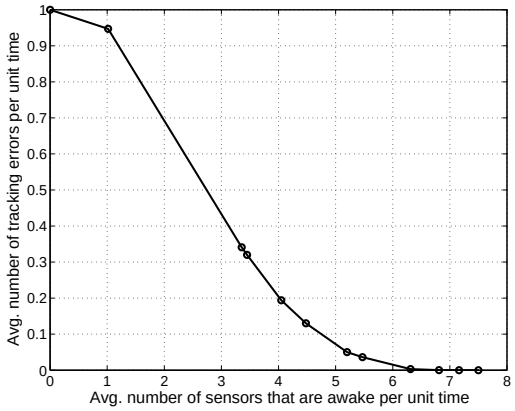

Fig. 3. Trade-off curve for the minimum sleep time policy for tracking an object undergoing random walk with $w_k = \{-1, 0, 1\}$ with equal probability ($b = 20$).

average tracking error approaches one and the average number of sensors that are awake approaches zero.

On the other hand, when c is large compared to the average number of sensors that has to be awake for the "always track" policy, the average tracking error approaches zero and the sleeping policies tend to the "always track" policy.

The average number of tracking errors is a non-increasing function of the average energy spent by the system.

The locally-optimal and minimum sleep time policies perform very similarly.

In the case of tracking a symmetric random walk allowing 10% tracking error, results in a 25% saving in energy. In the case of random walk with $w_k = \{-1, 0, 1\}$ with equal probability, the same 10% tracking error results in a 37% saving in energy. For a realistic tracking problem where the variation in velocity may be even higher, we expect to save even more energy by allowing a small tracking error.

9. CONCLUSION

We considered the problem of tracking an object undergoing a random motion in a field of sensors, where the sensors are put into sleep mode to conserve energy. We assumed that the sensors cannot be woken up prematurely from the sleep state. To obtain design guidelines for the sleep policy, we first considered a simplified problem of one-dimensional sensor network with uniformly spaced sensors, tracking an object undergoing a symmetric random walk. We showed that computing and implementing the optimal sleeping policy is intractable for this problem, even for a network with only seven sensors. We then developed a locally-optimal policy that can be simulated for this simplified problem. However, we argued that locally-optimal policy is impractical for implementation even for this simple scenario, and furthermore, it cannot be simulated for more complex tracking scenarios. Finally, we developed the minimum sleep time policy, which is tractable and can easily be extended to more complex scenarios. We argued that this policy should perform close to the locally-optimal policy when the average number of tracking errors is small. We verified this conjecture via simulations. Our simulations also show that significant savings in energy can result by allowing for some tracking error.

In our analysis we assumed the presence of a central unit that determines that sleep times for the sensors. It is of interest to study the more complicated decentralized case where the sensors that are awake communicate directly with each other without a central controller.

10. REFERENCES

[1] Wensheng Zhang and Guohong Cao, "DCTC: Dynamic Convoy Tree-based Collaboration for target tracking in sensor networks," *IEEE Transactions on Wireless Communications*, vol. 3, no. 5, pp. 1689–1701, September 2004.

[2] Wensheng Zhang and Guohong Cao, "An energy efficient framework for mobile target tracking in sensor networks," in *Military Communications Conference*, 2003, vol. 1.

[3] R. R. Brooks, P. Ramanathan, and A. M. Sayeed, "Distributed target classification and tracking in sensor networks," *Proceedings of the IEEE*, vol. 91, no. 8, pp. 1163–1171, August 2003.

[4] S. Balasubramanian, I. Elangovan, S. K. Jayaweera, and K. R. Namuduri, "Distributed and collaborative tracking for energy-constrained ad-hoc wireless sensor networks," in *Wireless Communications and Networking Conference*, 2004, vol. 3.

[5] R. Gupta and S. R. Das, "Tracking moving targets in a smart sensor network," in *Wireless Communications and Networking Conference*, 2004, vol. 3.

[6] Yingqi Xu and Wang-Chien Lee, "On localized prediction for power efficient object tracking in sensor networks," in *Proceedings of the 23rd International Conference on Distributed Computing Systems*, 2003.

[7] H. Yang and B. Sikdar, "A protocol for tracking mobile targets using sensor networks," in *Proceedings of the first IEEE International Workshop on Sensor Network Protocols and Applications*, 2003.

[8] M. Srivastava, "Challenges in energy-aware networked sensor systems," in *Proc. NSF Workshop on Networked Sensor and Actuator Systems*, Sept. 2003, http://www.cens.ucla.edu/Events/nsf/index.html.