Energy Efficient Multiple Target Tracking in Wireless Sensor Networks

Wai-Leong Yeow, Student Member, IEEE, Chen-Khong Tham, Member, IEEE, and Wai-Choong (Lawrence) Wong, Senior Member, IEEE

Abstract—Energy awareness is a crucial component in the design of wireless sensor networks at all layers. This paper looks into efficient energy utilization of a target-tracking sensor network by predicting a target’s trajectory through experience. While this is not new, the chief novelty comes in conserving energy through both dynamic spatial and temporal management of sensors while assuming minimal locality information. We adapted our target trajectory model from the Gauss–Markov mobility model, formulated the tracking problem as a hierarchical Markov decision process (HMDP), and solved it through neurodynamic programming. Our HMDP for target-tracking (HMTT) algorithm conserves energy by reducing the rate of sensing (temporal management) but maintains an acceptable tracking accuracy through trajectory prediction (spatial management) of multiple targets. We derived some theoretical bounds on accuracy and energy utilization of HMTT. Simulation results demonstrated the effectiveness of HMTT in energy conservation and tracking accuracy against two other predictive tracking algorithms, with accuracy of up to 47% higher and energy savings of up to 200%.

Index Terms—Distributed tracking, intelligent sensors, minimum energy control, neurodynamic programming, prediction methods.

I. INTRODUCTION

MANY CHALLENGES surface when tracking mobile targets effectively and efficiently in an unreliable sensor network. Readings from sensors with overlapping detection regions may be inconsistent due to noisy sensors or transmission errors and smart data fusion algorithms are required to disambiguate these irregularities [1]. Precise target classification and detection [2] from fused sensor readings is also mandatory. Furthermore, energy-efficient data routing protocols [3], [4] should be used.

In our approach, we anchor the target-tracking problem at a higher layer where sensor clusters collaboratively take turns to track a mobile target based on predictions through a tracking model learned through experience. This is analogous to Zhao’s information driven sensor query (IDSQ) [5], where querying sensors collaborate to increase target information while reducing communication cost. Our approach conserves energy by having tracking sensors work minimally and nontracking sensors sleep, since sleeping is the most fundamental and effective energy conservation strategy. The effectiveness of energy conservation achieved through sleeping can be illustrated via Berkeley MICA2 motes [6], where current consumption drops a few thousand fold when sleeping. Ye [7] devised a collaborative sleep–awake strategy to lengthen the lifespan of a sensor network. However, ensuring 100% sensor coverage is inefficient for target tracking. Sensors should only be activated when the target is in their vicinity, i.e., along the target’s trajectory. Consequently, if the target’s movement patterns are entirely known, sensors could be programmed to activate only when required. Xu et al. [8] tried to predict the target’s movements and activate only the required sensors through a prediction-based energy-saving scheme (PES). However, PES assumed perfect localization and straight-line trajectories (since no explicit mobility models were defined). This is unlike [9] and [10], where mobility models were defined for location tracking in personal-communications-service (PCS) and asynchronous-transfer-mode networks.

We propose a more realistic and novel mobility model in this paper and devise an energy-conserving method based on this model for sensor management called hierarchical Markov decision process (HMDP) for target tracking (HMTT). HMTT further conserves energy by dynamically adapting the sleep time of sensors, depending on the state of the detected targets. HMTT also has the advantage of “memorizing” target trajectories that is vital in target tracking. In particular, since sensors are generally small devices with limited resources, our method does not impose a high computational and memory load onto the sensors because the time and space complexity of HMTT is low enough for existing sensor nodes such as MICA2. The main contributions of this paper include the following:

1) a new target mobility model and a corresponding target-tracking model for energy efficient target tracking;
2) an energy efficient cluster-based target-tracking methodology using a two-level MDP formulation without assuming fine-grained localization;
3) a two-level hierarchy MDP algorithm to predict target trajectories and determine optimal sleep time of sensors in terms of energy utilization and prediction accuracy;
4) time and space complexity analysis of HMTT;
5) theoretical lower bound of energy utilization and upper bound of prediction accuracy of a cluster;
6) performance evaluation of HMTT via simulations and comparisons to other predictive tracking algorithms.

The rest of this paper is organized as follows: In Section II, the sensor network and target mobility and tracking models are explained. In Section III, the energy-efficient target-tracking...
problem is formulated as an HMDP. In Section IV, the theoretical background of the $Q(\lambda)$-learning algorithm and an analysis of its time-complexity are presented. In Section V, a detailed solution using $Q(\lambda)$, which forms our proposed HMTT algorithm and an analysis of the performance bound of HMTT, are also presented. Section VI shows the simulation results, and Section VII concludes this paper.

II. TARGET-TRACKING SENSOR NETWORK MODEL

We assume a sensor network as shown in Fig. 1. Sensors are uniformly distributed in a 2-D map and are grouped into clusters with some sensors as cluster heads. Generally, clustering [11], [12] improves scalability and facilitates localized algorithms for easy coordination [13]. Each sensor has a region of detection with radius $R$ and measures only the velocity $(\hat{v}, \hat{\theta})$ of detected targets. The absolute target position is unknown since sensors are not localized. The time required to wake up and sense is assumed to be a constant $t_w$ seconds. Thus, varying the sensors’ sleep time $\Delta t$ gives a dynamically varying duty cycle.

There are three states that correspond to a cluster’s alertness: sensing, listening, and tracking. In the sensing state, a cluster is on low alert and its sensors operate on a default duty cycle (sleep time $\Delta t$) to scan the area for targets. On detection of $\kappa_t$ targets at time $t$, all sensors in the cluster go into high alert and assume tracking state. Given readings from its sensors, the cluster head attempts to predict in which location area $\bar{L}_k$ a target $k$ will land in the next $t_w$ seconds and warns the other relevant clusters to prepare for all targets at time $t$. Thus, the predicates for listening and sensing states are then $(\kappa_t = 0) \lor (\kappa_{t-\Delta t} > 0)$. Consequently, the predicates for listening and sensing states are then $(\kappa_t = 0) \land (\kappa_{t-\Delta t} > 0) \land (\varpi_t > 0)$ and $(\kappa_t = 0) \land (\kappa_{t-\Delta t} > 0) \land (\varpi_t = 0)$, respectively. Note that these state definitions, when applied to single targets, are equivalent to the single target formulation [15], where $\kappa_t \in \{0, 1\}$, and $\varpi_t \in \{0, 1\}$.

A. Mobility Model

The area of surveillance is assumed to be broken up into nonoverlapping location areas. The rationale is that most tracking applications are interested in where locations targets are and not the exact Cartesian coordinates. Therefore, we are only interested in retrieving the sequence of location areas of a target $k : \{L_k(t)\}$, eliminating the need for strong localization. Cluster heads only need to know their respective location area that are achievable through reference beacons. The target movement model builds on this concept, as shown in Fig. 3. The specific target dynamics $x_t$, where $x_t = v_t$ or $\theta_t$ depends on its current location area $L$ and is adapted from a Gauss–Markov (GM) process [9]:

$$x_{t+\Delta t} = \rho \sigma \frac{x_t}{t} + (1 - \rho) \mu + z \sqrt{1 - \rho^2} \quad (1)$$

where $z \sim N(0, \sigma^2)$, and $\rho$ measures the degree of continuity s.t. $\rho$ is a function of $\Delta t$, and $L$ and $R_{xx}(\Delta t) = \rho \sigma^2$, $\mu$ and $\sigma^2$ denote the mean and variance of $x$, respectively.

The GM mobility model was introduced by Liang and Haas [9] for predictive mobility management in PCS networks, but targets under this model do not exhibit any interest in their destinations. On the other hand, the model of Liu et al. [10] addresses this, but it is defined as a deterministic mobility pattern at the higher level. Our model is stochastic and both location and destination dependent, since $P(L'\mid L)$ is a measure over the mean direction $\mu_\theta$.

Target interarrival times follow a known stochastic distribution and are location area specific. In Section V, they are assumed to be exponentially distributed with mean $t_w$ seconds. This does not limit the applicability of the algorithm because the formulation is generic enough to be applied for other cases.

B. Tracking and Prediction Model

We propose a simple but generic trajectory prediction model to predict targets such as those described above. Although optimal tracking algorithms such as Kalman filters [16] exist, they cannot be used because target mobility models, and their parameters are assumed known. This is the same for IDSQ [5]. For the GM model, estimation techniques such as linear minimum mean-squared error (LMMSE) [17] can be used.

A simpler method to LMMSE is parameter estimation. Let $x_t$ be the kinetic state of the GM process (1) to be estimated.

---

1Current localization methods are largely inaccurate [14]. Hence, in our formulation, targets are tracked with respect to the weak localization information, as explained later.
Since \( x_t \) is wide-sense stationary and \( z \) is Gaussian with mean 0, the maximum-likelihood estimate of \( x_{t+\delta t} \) is then
\[
\hat{x}_{t+\delta t} = \hat{\mu}_x x_t + (1 - \hat{\rho}) \hat{\mu}_x
\]
where \( \hat{\mu}_x \) and \( \hat{\rho}_x \) are the estimated mean and autocorrelation coefficient, respectively, from \( N \) sequential samples of \( x_t \):
\[
\hat{\mu}_x = \frac{1}{N} \sum_{n=1}^{N} x_{n\delta t}
\]
\[
\hat{\rho}_x = \frac{\sum_{n=2}^{N} x_{n\delta t} x_{(n-1)\delta t}}{N - 1} - \left( \frac{\sum_{n=1}^{N} x_{n\delta t}^2}{N} \right) \delta t.
\]

Then, the position of the target \( p_i \) can be predicted via
\[
\hat{p}_{n\delta t} = p_{(n-1)\delta t} + \hat{v}_{n\delta t} \left( \cos(\hat{\theta}_{n\delta t}) \right) \delta t
\]
where \( p_{(n-1)\delta t} = p_0 + \delta t \sum_{m=1}^{n-1} v_{j\delta t} \left( \cos(\theta_{m\delta t}) \right) \). Unfortunately, \( p_0 \) cannot be determined unless the sensors are strongly localized. Hence, in our approach, targets are tracked at location areas rather than its position. A simple Markovian prediction model can be \( \hat{L}_k(t + \tau) = f_k(L, \tau_k) \), where \( f_k \) is some stochastic function, and \( \tau_k \) denotes the dwell time of target \( k \) in \( L \). Note that \( \tau_k = n\delta t \) if target \( k \) is detected for the \( n \)th time at sampling interval \( \delta t \). From (5), the predicted position \( \hat{p}_{n\delta t} \) depends on the kinetic state predictions \( \hat{\nu}_{n\delta t} \) and \( \hat{\theta}_{n\delta t} \). Since these states are Markovian, it is sufficient to base the target position prediction on \( \hat{\nu}_{(n-1)\delta t} \) and \( \hat{\theta}_{(n-1)\delta t} \). The generic prediction model on the next location area at time \( t \) for a small time interval \( \delta t \) then becomes
\[
\hat{L}_k(t + \delta t) = f_k(\nu_{k,t}, \theta_{k,t}, L, \tau_k).
\]

Note that this model generalizes over constant speed/turn models [16] and GM models and random walk models through a particular function \( f_k \), respectively. We use neurodynamic programming to learn the function \( f_k \) from history.

### III. Problem Formulation

Let the sleeping interval of listening state and tracking state be \( \Delta t_L \) and \( \Delta t_T \), respectively. For conventional tracking networks, it is logical to set \( \Delta t_s > \Delta t_L > \Delta t_T \) according to their alertness levels. However, since targets show preference in certain routes, clusters that lie along high interest areas will exhaust faster than other clusters [8]. We use prediction to solve this. Suppose target trajectories are known; then, only clusters in \( \{ L_k(n\delta t) : n \in \mathbb{N} \} \) should be activated for some reporting rate \( \Delta t \), achieving energy efficiency. Hence, we rely on learning trajectories and then deciding appropriate sleep times \( \Delta t \) for energy efficiency.

We assume the following.

1) Target trajectories are Markovian and can be segmented into independent location areas \( L \) (See Section II-A).
2) Targets are segregated into different classes, and targets of the same class possess the same trajectory parameters.

For simplicity, we have only considered one class of targets. Section V explains the extension of this algorithm to the multiple classes and targets scenario.

We formulate the energy efficient tracking problem as a distributed HMDP [15]. Global tasks are divided into \( \mathcal{O} \) sub-tasks to be carried out by \( \mathcal{O} \) lower level agents (LLAs). The higher level agent (HLA) chooses a subtask among \( \mathcal{O} \) LLAs at each time step \( i \) to optimize some overall global cost function \( J \). Control is then passed to the chosen LLA \( o \), which, in turn, optimizes a local cost function \( J_o \) over interval \( i \). In our problem, two tightly coupled actions are to be determined at each state: 1) duty cycle of the sensors and 2) predictions of target trajectories when tracking. These are interrelated since the longer the \( \Delta t \) of the sensors, the further the prediction \( \hat{L}_k \) of targets would be. We decompose the problem into a hierarchy of two levels for each action. This way, two advantages can be observed: 1) Modularity is ensured while decoupling the two decision processes, and 2) the state-action space is reduced, resulting in faster convergence of a solution.

HLA acts on a slower time scale \( i \) where the sleeping time \( \Delta t \) is determined. The immediate cost \( g_i \) is given by
\[
g_i = \alpha \frac{P_i}{P_{\text{max}}} + (1 - \alpha) \sum_{k=1}^{N_{\text{ targets}}} X_{k,i} n_{k,i}
\]
where \( \alpha \) is a system parameter that balances the conflicting goals between power used \( P_i \) and number of wrong predictions \( X_{k,i} \) for target \( k \) in interval \( i \) (i.e., from \( t_{i-1} \) to \( t_i \)). \( P_{\text{max}} \) is the maximum power that can be consumed if the fastest sensing
interval is used (i.e., smallest $\Delta t$ chosen), and $N_{k,i}$ is the number of predictions for target $k$ made in interval $i$.

In order to optimize tracking accuracy and energy consumption, the expected sum of discounted costs $J = E[\sum_{t=0}^{\infty} \gamma_t g_t]$ toward an infinite horizon is minimized. This is a popular performance metric that is used intensively in MDPs. The other reason for using discounted costs is motivated by the virtue of the problem. Due to limited battery life, clusters cannot operate infinitely. It is desirable to make the clusters myopic and act on a discounted horizon. This is achieved via discounting immediate costs as opposed to optimizing average immediate costs (the other popular performance metric). Although dynamic programming (DP) has been studied extensively for finite horizon cases as well [18], finite-horizon DP cannot be applied directly to our problem because in minimizing immediate costs, the overall lifetime is affected and does not stay constant. We verify via simulations as shown in Section VI that the expected sum of discounted costs is a good optimization metric

$$J = \alpha E \left[ \sum_{t=0}^{T} \gamma_t \frac{P_t}{P_{\text{max}}} \right] + (1-\alpha) E \left[ \sum_{t=0}^{T} \gamma_t \frac{\sum_{k=1}^{n_{k,i}} X_{k,i}}{\sum_{k=1}^{n_{k,i}} N_{k,i}} \right].$$

The significance of $\alpha$ shows up in the optimization function (8) that weighs between the discounted sum of power consumed and the discounted sum of erroneous predictions.

Higher level state space is defined as a two-tuple space: $(L, t_w)$. In the listening state, $L$ is the location of the tracking cluster that issued the alert of an incoming target in $t_w$ seconds. In the tracking state, $L$ is the location of the current cluster with $t_w=0$, and $(0,0)$ represents sensing state. Hence, if $L$ is the total number of location areas in the surveillance area and $W$ is the number of values $t_w$ can assume, then there are $W(N-1)$ listening states because targets behave uniquely at different location areas and time instances.

LLA is the agent that acts on subtasks (characterized by $\Delta t$) chosen by HLA. These LLAs act within HLA’s tracking state on a faster time scale $n$ to perform trajectory prediction at every $\Delta t T + t_u$ seconds. LLA implements the function $f_k$ in (6). Thus, its state space is described by ($\hat{\nu}, \hat{\theta}, \tau, \Delta t$), and its immediate cost $\zeta_{i,n}$ at the HLA interval $i$ is

$$\zeta_{i,n} = \begin{cases} 0, & \text{correct prediction} \\ 1, & \text{otherwise.} \end{cases}$$

Consistent with HLA, the total expected cost $J_i = \sum_{n=0}^{\tau_i/\Delta t} \gamma_i^t \zeta_{i,n}$ is minimized, where $\tau_i$ is the duration in which the cluster stayed in the tracking state for interval $i$, and $\gamma_i$ is a similar parameter to $\gamma_n$, where $\gamma_n, \gamma_i \in (0,1)$. Note that LLAs are pure trajectory predictors and could be substituted by any method discussed in Section II-B.

In the event that a wrong prediction is made, the cluster on which the target landed is unwarned and would persist in its state. Hence, it is possible that this target may be lost, and there is no way to recover it unless a network-wide search is conducted. For energy efficiency, such searches are suppressed, and it is left entirely to other clusters to detect the target as a newly appeared target.

IV. SOLVING MDP THROUGH $Q$-LEARNING

We use $Q$-learning to solve the two-level MDP, as formulated in Section III. $Q$-learning has the advantage of solving MDP without any knowledge of state transition probabilities and escapes the curse of dimensionality in MDPs through learning. The former is particularly useful in the target-tracking scenario, where target dynamics are unknown, and optimal actions (sensing interval and predictions) are determined through learning. This section gives a brief introduction of $Q$-learning. Although $Q$-learning is not intended to solve HMDP problems, we show in Section V how it can be modified slightly for the energy-efficient target-tracking problem and, in Section V-C, the proof of convergence of the modified algorithm.

A. Bellman’s Optimality Equations

Let $S$ be a finite set of states and $A(s)$ be a set of finite actions permissible in state $s \in S$. These states and actions correspond directly to that of HLA and LLA. In each state $s$, an agent chooses an action $a \in A(s)$ from a stationary policy $\pi$, where $\pi(s) = a$ and gets an immediate cost $g(s,a)$. The system then transits into a new state $s' \in S$. Bellman’s equations [18] which solve for the optimal policy $\pi^*$ for MDPs that yield the minimum expected sum of $\gamma$-discounted costs $J$ are defined in:

$$\pi^*(s) = \arg \min_{a \in A(s)} Q(s,a)$$

$$J^*(s) = \min_{a \in A(s)} Q(s,a)$$

$$Q(s,a) = g(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) J^*(s').$$

Solving Bellman’s equations is usually computationally intensive because both the time and space complexity grows exponentially with the dimensions of states and actions. This is known as the curse of dimensionality.

B. $Q(\lambda)$ Algorithm

There are several ways to solve Bellman’s equations: linear programming, policy iteration, value iteration, etc. These methods require the right-hand side of (12) to be known, i.e., $g(s,a)$ and $P(s'|s,a)$, $g(s,a)$ is usually known since it is part of the problem definition. Unfortunately, the state-transition probabilities $P(s'|s,a)$ is often an unknown entity that depends on the environment in which the agent works.

$Q(\lambda)$ [19] is an iterative algorithm that, without prior knowledge of $P(s'|s,a)$, approximates the optimal solution through simulation or implicit sampling. At each state $s_{i+1}$ of iteration $i+1$, it updates $\hat{Q}_{i+1}(s_i, a_i)$, which is an estimate of the $Q$-function, by computing the estimation error $\delta_i$ after incurring a cost of $g(s_i, a_i)$ in the previous iteration. From the updated $Q$-function estimate, it chooses the next course of action $a_{i+1}$.

Watkins’ $Q$-learning algorithm updates $Q(s,a)$ by

$$\hat{Q}_{i+1}(s_i, a_i) = \hat{Q}_i(s_i, a_i) + \alpha_i \delta_i$$

Authorized licensed use limited to: Bangladesh Univ of Engineering and Tech. Downloaded on May 11, 2009 at 05:29 from IEEE Xplore. Restrictions apply.
where $\alpha_i \in (0, 1)$ is the learning rate which decreases with $i$. One suitable learning rate could be $\alpha_i = \eta/i$, with $\eta \in (0, 1)$. $\delta_i$ is evaluated by

$$\delta_i = r(s_i, a_i) + \gamma \hat{J}_i(s_{i+1}) - \hat{Q}_i(s_i, a_i)$$

$$\hat{J}_i(s_{i+1}) = \min_{a \in A(s_{i+1})} \hat{Q}_i(s_{i+1}, a)$$

where $\hat{J}_i(s)$ is an estimate of the expected sum of discounted costs $J$ in (8) with initial state $s$. The estimated policy is then

$$a_i = \pi_i(s_i) = \arg \min_{a \in A(s_i)} \hat{Q}_i(s_i, a).$$

$Q(\lambda)$ improves $Q$-learning through eligibility traces $e(s, a)$. Here, $\lambda$ is another learning parameter similar to $\alpha_i$ that has to be in the range of $[0, 1]$ for guaranteed convergence. In general, eligibility traces give a higher update factor for recently revisited states. This means that the eligibility trace for a state-action pair $(s, a)$ will be reinforced if $s = s_i$ and $a = a_i$. Otherwise, if the previous action $a_i$ is not greedy, the eligibility trace is cleared. The new update equation for $Q(s, a)$, $\forall a \in A(s)$, $\forall s \in S$, is then changed from (13) to

$$\hat{Q}_{i+1}(s, a) = \hat{Q}_i(s, a) + \alpha_i \delta_i e_i(s, a)$$

where

$$e_i(s, a) = \mathcal{I}_{sx} \mathcal{I}_{ax} + \left\{ \begin{array}{ll} \gamma \lambda e_{i-1}(s, a), & a_i = \pi_{i-1}(s_i) \\ 0, & \text{otherwise} \end{array} \right.$$}

and $\mathcal{I}_{xy}$ is an identity indicator function, which equals to 1 if $x = y$ and 0, otherwise.

C. Time and Space Complexity of the $Q(\lambda)$ Algorithm Using a Cerebellar Modulation Articulation Controller (CMAC)

CMAC [20] is a type of neural network that is used in our implementation to approximate the $Q$-function. It has the advantages of 1) interpolating unseen values of $Q(s, a)$, thereby speeding up learning further, and 2) being fast with access time $D = O(1)$.

We analyze the time complexity of obtaining an action $a$ through $Q(\lambda)$ at each state $s$, which consists of executing (16) and updating the $Q$-function in (17). The time complexity of the former task is usually small (using either CMAC or lookup tables): $O(|A(s)|)$. If tree structures are used, then this can be further reduced to $O(\log |A(s)|)$, but that would increase the access time $D$ from $O(1)$ to $O(\log |A(s)|)$.

The latter task is much more computationally intensive ($O(D |S||A|)$) since every state-action pair $(s, a)$ is traversed. However, this can be reduced significantly if we truncate eligibility traces. From (18), eligibility traces are only nonzero either when 1) a greedy action was taken at the previous iteration $i - 1$, i.e., $a_i = \pi_{i-1}(s_i)$, or when 2) $(s, a)$ corresponds to the state-action pair $(s_i, a_i)$ at iteration $i$. Thus, nonzero eligibility traces can be stored as a list, terminating at the last iteration $j$ when a greedy action was taken, i.e., $e(s_i, a_i), e(s_{i-1}, a_{i-1}), \ldots, e(s_j, a_j)$. Moreover, eligibility traces are discounted by a factor of $(\gamma \lambda)^C$ for $C$ iterations away. It is, thus, safe to assume $e(s_{i-c}, a_{i-c}) = 0$ for all $c$ such that $C < c < j$ in a simplified, practical implementation. Then, the update task only traverses through a constant number of state-action pairs $C$, reducing the time complexity to $O(D)$. This gives the overall time complexity to obtain $a$ through $Q(\lambda)$ to be $O(|A(s)|)$.

The space complexity of a CMAC is $\Theta(qd^2)$, where $r$ is the resolution of each tile, $q$ is the number of quantizers, and $d$ is the dimension of the input space $S \times A$. For a two-quantizer CMAC, this evaluates to 240 kbytes, which can be fitted into a Berkeley mote’s serial flash.

Note that $Q(\lambda)$ is an online algorithm that learns the optimal policy through sampling the environment. If the model of the environment $P(s'|s, a)$ is known, optimal policies can be computed offline prior to deployment. Then, the memory requirement can be reduced to 40 kbytes, and time complexity can be reduced to $O(1)$ via a lookup table-based implementation, eliminating the need to update estimates of the $Q$-function.

D. Relationship With Other HMDP Formulations

Our HMDP formulation differs from most well-known HMDP formulations because the higher level acts on a different time scale as the lower level. The HMDP formulation by Chang et al. [21] is an exception: Their multi-time scale MDP (MMDP) captures the difference in time scales at different levels. Furthermore, Chang et al. proposed a method of solving the multi-time scale problem and proved its convergence. Our formulation differs from that of MMDP in that the LLA time scale varies according to the subtask chosen, and the HLA chooses the subtask by selecting the appropriate LLA sleeping time at each sensing cycle. The higher level cost formulation also does not follow that of MMDP but follows more closely to Tham’s composite $Q$-learning [20]. Thus, $Q$-learning is proposed rather than Chang’s method for solving the HMDP. The other motivating reason is that $Q$-learning does not assume a priori knowledge of state transition probabilities, which is required in Chang’s solution.

Another distinct difference in our hierarchical approach to that of Chang’s MMDP is that LLA is only in action when HLA is in tracking state, since LLA is a trajectory predictor. The time scale at which LLA works at depends largely on HLA and is not constant at each time step. HLA’s time scale is also largely dependent on the target states. These considerations are captured in Sutton’s options formulation [22] instead.

An option $o = (\mathcal{I}, \pi, \beta)$ is introduced where the HLA’s action is to choose over a set of options $O$, $\mathcal{I}$ is the set of states where $o$ is permissible. The LLA then assumes control with policy $\pi$ until termination condition $\beta$. A direct mapping then follows.

1. $\mathcal{I} = \{(L, 0)|L\}$ is the location of the current cluster.
2. $\beta$ is true if and only if no target is detected in $L$.
3. $\pi$ is the LLA policy under a HLA chosen $\Delta t$.

Both formulations by Chang and Sutton are proven to converge. We present our algorithm HMTT in the next section and the proof of its convergence in Section V-C.
Fig. 4. Example of a target moving after $3\delta t$ seconds.

V. HMTT ALGORITHM AND ANALYSIS

A. Higher Level Agent (HLA)

HLA determines the sensing interval ($\Delta t + t_a$). In the sensing state, since target interarrival times are exponentially distributed, it is unjustifiable and inefficient to learn the optimal $\Delta t$ using $Q(\lambda)$. Moreover, missed targets cannot be characterized since no such feedback is possible. Instead, if we desire $P$(missing target) $< p$, then $\Delta t$ can be found by solving $P(t_a < \Delta t) < p$ [15], which gives

$$\Delta t < -\bar{t}_a \ln(1 - p)$$

where $t_a$ is the difference between the time when the new target appears and the time when the previous target(s) leaves the location area, and $\bar{t}_a$ denotes the mean of $t_a$. In other words, we hope to detect the target when it appears immediately. The value of $\Delta t$ obtained from (19) is generally small because it ignores the fact that a target is still detectable so long as it is still in the same location area. Taking dwell time $\tau$ into consideration, a better $\Delta t$ of longer duration can be calculated using

$$P(t_a \leq \Delta t < t_a + \tau) > 1 - p.$$  

However, the probability distribution of $\tau$ is difficult to analyze because it involves analyzing the underlying GM process of $v$ and $\theta$ and taking into account the location area of radius $d$. To simplify analysis, consider a target that first appears in a location area in Fig. 4 where $\rho_0$ is assumed to be large. Then

$$P(\tau > n\delta t) \approx P(d_1 + d_2 + \cdots + d_n < d)$$

where $d_i = v_i \delta t$, for $i = 1, \ldots, n$. $v_i$ can be recursively expanded from (1) to give

$$v_i = \rho^i v_0 + (1 - \rho^i)\mu_v + \sqrt{1 - \rho^i} \sum_{j=0}^{i-1} \rho^{i-j-1} z_j$$

where $\rho_v$ is abbreviated as $\rho$. Then, the displacement is

$$D_n = \delta t \cdot \sum_{i=1}^{n} v_i$$

$$= \delta t \left[ \frac{1 - \rho^n}{1 - \rho}v_0 + \left( n - \frac{1 - \rho^n}{1 - \rho} \right)\mu_v + \sqrt{1 - \rho^n} \sum_{i=1}^{n} \sum_{j=0}^{i-1} \rho^{i-j-1} z_j \right].$$

The approximate cumulative density function of $\tau$ is then

$$F_\tau(n, \delta t | \mu_v, \sigma_v, \rho, d) = \begin{cases} 1 - Q\left(\frac{d - \mu_{D_n}}{\sigma_{D_n}}\right), & n \in \mathbb{Z}^+ \\ 0, & \text{otherwise} \end{cases}$$

where $Q(x) = \frac{1}{2} \text{erfc}(x/\sqrt{2})$, and (20) becomes

$$1 - p < P(\tau > \Delta t - t_a, t_a \leq \Delta t)$$

$$= \sum_{i=1}^{m} P((i-1) \delta t < t_a \leq i \delta t) Q\left(\frac{d - \mu_{D_{n-i}}}{\sigma_{D_{n-i}}}\right)$$

where $m = \lceil \Delta t / \delta t \rceil$. We solve the above equation empirically. Table II shows the desired values of $p$, their corresponding $\Delta t$, and the actual values of $p$ found through simulations. Fig. 5 shows a graph of the simulated and calculated values of the left-hand side of (27) against $\Delta t$ for a cluster of radius 300 m, mean velocity $\mu_v$, 20 m/s, and mean arrival time $\bar{t}_a$ 10 s. As expected, the values of $\Delta t$ obtained through (27) are significantly larger than that of (19) (labeled “$\tau$ ignored”). This difference is approximately 14 s, which is at least twice the largest $\Delta t$ found through (19). Although the better performing method requires some prior knowledge of target velocity $v$, this method should still be used by assuming some moderate values of $\mu_v, \sigma_v, \rho$, and $\theta$ for a better result.

<table>
<thead>
<tr>
<th>$p$</th>
<th>calculated</th>
<th>simulated</th>
<th>$\tau$ ignored</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.22</td>
<td>16.2</td>
<td>16.5</td>
<td>2.48</td>
</tr>
<tr>
<td>0.32</td>
<td>19.1</td>
<td>17.9</td>
<td>3.86</td>
</tr>
<tr>
<td>0.40</td>
<td>20.5</td>
<td>19.1</td>
<td>5.11</td>
</tr>
<tr>
<td>0.50</td>
<td>22.3</td>
<td>20.8</td>
<td>6.93</td>
</tr>
</tbody>
</table>

Fig. 5. Detection probability against sleeping interval $\Delta t$.
In the other states (tracking and listening), HLA determines the appropriate $\Delta t$ through $Q$-learning described in Section IV. The states $s$ are in the form of $(L, t_w)$, as defined in Section III and the immediate cost $g(s, a)$ is equal to $g_i$, as defined in (7). The various permissible values of $\Delta t$ are then directly mapped from the different actions $a$ from which the Q-learner chooses.

Fig. 2 shows the possible state transitions of an HLA. At each new state $s_{i+1}$, HLA obtains an immediate cost $g_i$, which serves as an indication of the amount of energy spent and the prediction accuracy of LLA in the time step $i$ resulting from a previous action $a_i$. HLA updates its $Q$-function estimate using (17) to learn the best $\Delta t$ to choose for a given state $s$. The “optimal” $\Delta t$ for the current time step $i$ can then be determined through (16). However, this “optimal” action is not always executed, due to the exploration-exploitation dilemma in reinforcement learning problems [19]. This is to ensure an adequate amount of exploration is done while exploiting the current “optimal” action to avoid converging at local minima. In our implementation, we use an $\epsilon$-greedy scheme where the probability of not taking the current “optimal” action is $\epsilon$. $\epsilon$ is decreased with time to ensure convergence.

B. LLA (Trajectory Predictor)

When the cluster enters the tracking state and starts sensing at every $\Delta t + t_w$ seconds, control is passed to LLA for trajectory prediction. LLA works in discrete intervals at the beginning of every sensing cycle with the immediate costs $s_{i,n}$ defined in (9) and the states and actions defined in Section III for each detected target $k$. For the duration $\tau_{k,i}$ at which a target $k$ stays in the current location area $L$, LLA will execute on a finite horizon of $l$ stages, where $l = [\tau_{k,i}/\Delta t]$. Here, we make a slight modification to the MDP on which LLA executes to avoid the problem of having nonstationary optimal policies for finite horizon MDP [23] (furthermore, $l$ is stochastic and depends on $\tau_{k,i}$). An absorbing start/termination state $s_T$ is inserted such that all LLAs begin with state $s_T$ when any target $k$ enters the current location area $L$ and terminates with state $s_T$ when it leaves. This transforms the original finite horizon MDP to an infinite horizon MDP [24].

If LLA predicts that target $k$ will end up in a neighboring location $L_k \neq L$ at the $n$th time step, LLA send prediction messages to clusters in $L_k$ with $t_w = \Delta t$. If this prediction is correct, target $k$ is detected within $t_w$ seconds in $L_k$, these clusters in $L_k$ send acknowledgment messages to LLA. Thus, LLA’s immediate cost $s_{i,n}$ is 0 so long as one acknowledgment is received. Otherwise, LLA assumes a wrong prediction, and the immediate cost $s_{i,n}$ is 1. Thus, negative acknowledgments need not be sent in order to conserve more energy. When target $k$ leaves $L$, LLA transits into the absorbing termination state $s_T$ with an immediate cost of 0 and returns control to HLA.

Since multiple targets might arrive at the cluster, $k_t$ predictions would be made by $k_t$ LLAs running in parallel. In our simulations, we consider the case where targets all follow a similar mobility model with the same parameters. Hence, trajectory sequences could be jointly updated into the same $Q$-function as in (17). For the more general and trivial case where each target conforms to one of $\zeta$ mobility models, we have $\zeta Q$-functions at LLA. If all $k_t$ detected targets observe distinct mobility models, then the HLA immediate cost $g_i$ should be amended as

$$g_i = \alpha \frac{P_i}{P_{\text{max}}} + (1 - \alpha) \max_{k < \zeta} X_{k,i} \frac{\lambda_k}{N_{k,i}}.$$  (28)

For the case where some of the $k_t$ targets observe the same mobility model $\zeta$, these targets may be grouped together. Hence

$$g_i = \alpha \frac{P_i}{P_{\text{max}}} + (1 - \alpha) \max_{\zeta} \sum_{k < \zeta} X_{k,i} \frac{\lambda_k}{N_{k,i}}.$$  (29)

The estimated LLA policy $\hat{\pi}$ through $Q$-learning can be seen as a direct implementation of function $f$ in (6). For any target $k$ which spends $\tau_k$ seconds in $L$ with velocity $(v_k, \theta_k)$ at $t$, the location $L'$ of target $k$ at time $t'$ is found to be

$$L' = \hat{\pi}(v_k, \theta_k, \tau_k, t' - t).$$  (30)

We remark that the $Q$-function used to evaluate the LLA policy from (16) implicitly “stores” target trajectories. If ample exploration is performed, the $Q$-function is the expected sum of discounted probability of occurrences of the target, given its velocity, direction, and dwell time. This information may be useful for applications that are interested in retrieving a summary of the trajectories of detected targets.

C. Convergence of Hierarchical $Q(\lambda)$

We present the proof of convergence of our hierarchical method (see Section IV) in this section. This proof relies on the fact that $Q(\lambda)$ converges to the optimal solution [19] from the lower level to the higher level MDPs.

Since $Q(\lambda)$ converges to the optimal solution at LLA, then the sum of discounted costs $J_i = \sum_{n} \gamma^i \widehat{s}_{i,n}$ converges to a minimum constant value. Further note that for each HLA interval $i$, $\sum_{i} \widehat{s}_{i,n} = \sum N_{k,i} - \sum X_{k,i}$. The term $\sum X_{k,i}$ is minimized under $Q(\lambda)$ because $\sum N_{k,i}$ is determined by the environment and is constant for some $\Delta t$. Thus, for any HLA interval $i$, $\sum X_{k,i} / \sum N_{k,i}$ tends to a minimum value $\chi^*$. Then, the HLA cost $g_i$ becomes

$$g_i = \alpha \frac{P_i}{P_{\text{max}}} + (1 - \alpha) \chi^*.$$  (31)

Transitions at LLA can then be treated as a one-step transition at HLA for listening and tracking states. This is analogous to Chang’s convergence proof of MMDP [21]. Thus, policies at these two HLA states converge.

In the sensing state where $Q(\lambda)$ is not used, $g_i$ is

$$g_i = \alpha \frac{P_i (\Delta t + t_w)}{P_{\text{max}} (\Delta t + t_w)}$$  (32)

where $P_i$ is the power consumed while sleeping, and $E_s$ is the energy used while sensing for targets. Since the default $\Delta t$ is a constant value, $g_i$ will also be unchanged. This implies that the policy for HLA at sensing state does not change, completing the proof of convergence of hierarchical $Q(\lambda)$. 

D. Theoretical Bounds

In this section, we derive the energy spent and tracking accuracy of a cluster (HLA), given perfect trajectory predictors (LLA). This gives the theoretical performance bound of the system. Accuracy is defined as the total number of correct predictions against total number of predictions, and the perfect predictor is defined as having perfect knowledge of $P_{k}(L', t + \Delta t| L, t)$, which is the probability a target $k$ moves to $L'$ after $\Delta t$ seconds after staying in $L$ for $t$ seconds. Hence, to maximize accuracy, the perfect trajectory predictor will choose $\hat{L}$ from neighborhood $N$ such that

$$\hat{L} = \arg \max_{L \in N} P_{k}(L', t + \Delta t| L, t). \quad (33)$$

Hence, the accuracy of the perfect predictor at time $t$ for the next $\Delta t$ seconds is given by

$$\Lambda_{k,L}(t, \Delta t) = P_{k}(\hat{L}, t + \Delta t| L, t). \quad (34)$$

Suppose the target spends $\tau_{k,i}$ seconds in $L$; then, LLA makes $N_{k,i} = \lceil \tau_{k,i} / \Delta t \rceil$ predictions, and the resulting average prediction accuracy is

$$\bar{\Lambda}_{k,L}(\tau_{k,i}, \Delta t) = \frac{1}{N_{k,i}} \sum_{n=1}^{N_{k,i}} \Lambda_{L} ((n - 1)\Delta t, \Delta t). \quad (35)$$

Since the predictor will not predict elsewhere for any time before $\tau_{i}$, i.e., $P_{k}(L', t + \delta| L, t) = 0$ when $t + \delta < \tau_{i}, \forall L' \neq L$, then (35) becomes

$$\bar{\Lambda}_{k,L}(\tau_{i}, \Delta t) = 1 - \frac{1}{N_{k,i}} \left[ 1 - P_{k}(\hat{L}, \tau_{k,i}| L) \right]. \quad (36)$$

Thus, for small $\Delta t$, the average accuracy tends to 1 for one target. For $K_{t}$ independent targets, the average accuracy across targets becomes

$$\bar{\Lambda}_{L}(\tau_{i}, \Delta t) = \frac{1}{K_{t}} \sum_{k=1}^{K_{t}} \bar{\Lambda}_{k,L}(\tau_{k,i}, \Delta t) \quad (37)$$

which still tends to 1 for small $\Delta t$.

Let the sensing energy consumption be $E_{s}$, sleeping power be $P_{s}$, and energy consumed for $\tau_{i}$ be $E_{g_{i}} \xi(\tau_{i})$. Let the minimum and maximum allowed $\Delta t$ be $\Delta t_{\text{min}}$ and $\Delta t_{\text{max}}$, respectively. For parameter $\alpha = 1$, only the energy component of $g_{i}$ is optimized, and for $\alpha = 0$, only accuracy is optimized. Hence

$$\xi(\tau_{i}) = \begin{cases} \left[ \tau_{i} / \Delta t_{\text{max}} \right] \cdot (P_{s} \Delta t_{\text{max}} + E_{g_{i}}), & \alpha = 1 \\ \left[ \tau_{i} / \Delta t_{\text{min}} \right] \cdot (P_{s} \Delta t_{\text{min}} + E_{g_{i}}), & \alpha = 0 \end{cases} \quad (38)$$

The theoretical bounds of average prediction accuracy and energy consumed are, thus, given by (37) and (38), respectively.

VI. Simulation Results

Our simulations are built using the simple-portable-simulation-language library [25] for discrete event scheduling. There are five sets of results.

1) Performance of HMTT against HLA discount factor $\gamma_{h}$ in a single target case. This serves to address the assertion made in Section III that cluster heads should be short-sighted and act on a discounted horizon.

2) Performance of HMTT against $\alpha$ in a single target case. This demonstrates the effect of $\alpha$ as in (8).

3) Performance comparison of HMTT on single targets to a) PES-INSTANT scheme [8], which assumes targets will continue with the same velocity for the next $\Delta t$ seconds. Unlike HMTT, PES cannot dynamically adapt $\Delta t$. This is our comparison baseline.

4) Performance of HMTT with a CMAC function approximator as LLA. This demonstrates the correctness of HMTT on multiple targets.

5) Performance comparison of HMTT on multiple targets with PES and HLPES again.

Performance evaluation is based on two metrics as follows.

1) Prediction accuracy refers to the average proportion of correct predictions across clusters.

2) Energy consumed refers to the total number of Joules consumed for one whole simulation.

The first three sets of simulations models a sensor network covering an area of interest of $200 \times 200$ m, which is divided into 14 location areas of $50 \times 50$ m and housing 14 clusters. There are routes whose targets are most likely to follow, and there are routes whose targets are not likely to follow. The targets dynamics are as follows:

1) velocity ranges from 0 to 30 m/s, with mean 15 m/s and standard deviation 15 m/s;

2) interarrival times follow an exponential distribution with $\bar{t}_{a} = 0.5$ s.

The energy consumption model used for simulations follows that of the Berkley MICA2 motes [6]: Transmitting and receiving a packet consumes 39.6 and 26.4 mW of power, respectively, and being in sleep and active modes consumes 0.04 and 42.9 mW of power, respectively. $\Delta t$ is selected in the range of $[0.5, 18]$ and sensing time $t_{s} = 0.1$ s. The default $\Delta t$ in the sensing state is set to 9 s after assuming the parameters above from (27) with $p = 0.4$.

Fig. 6 shows the maximum prediction accuracy and minimum energy consumed over 1000 simulation runs, each lasting 30 000 s. Performance is seen to drop drastically with a large $\gamma_{h}$ ($\geq 0.7$), where immediate costs $g_{i}$ decay more slowly over time. This agrees with our assertion in Section III that discounted costs should be used instead of the average cost criterion for MDPs.

In case 2), we set $\gamma = 0.5$ to show the effect of balancing parameter $\alpha$. One hundred fifty simulations are run for each $\alpha$ value, and their respective points are shown in Fig. 6(c). Each
Fig. 6. Simulation results for both single target [Fig. 6(a) and (b), case 1] and multiple targets [Fig. 6(c), case 2]. (a) Maximum accuracy against $\alpha$. (b) Minimum energy consumed against $\alpha$. (c) Effect of balancing parameter $\alpha$.

The group of points consists of 100 simulation runs, each lasting 80,000 s. For $\alpha = 0.1$, prediction accuracy reached 0.7, but the energy consumed increased as well to as high as 580 J. When $\alpha = 0.9$, energy consumed decreased to 570 J, but accuracy suffered at 0.54. For other $\alpha$ values, the respective convex hulls lie in between the two extremes in an orderly manner, showing that $\alpha$ is an effective control parameter to balance between the two conflicting goals. We note that the difference between 580 J and 570 J is not big. This is because the energy usage is summed across the whole sensor networks, including those which never entered the tracking state at all because the targets were never there. Hence, these clusters operate at the default sensing interval, which may not be the best strategy. This is true for other results as well.

The next set of simulations [case 3)] compares HMTT against two different sensing intervals (90% and 98% sleeping time) of the PES-instant scheme [8] and HLPES. Fig. 7 summarizes the performance of all four algorithms against $\alpha$ over simulations lasting 30,000 s each. HMTT can be seen to be the best in conserving energy while being effective in tracking for some $\alpha$ in the single target case. Prediction accuracy of HMTT ranges from 0.78 to 0.68. This is about 4.6% to 20% more than the next most accurate algorithm (PES—90%). In terms of energy conservation, HMTT is not the best algorithm, but it is better than both PES schemes.

The next set of simulations models another sensor network of 95 clusters in a 500 x 500 m region. The region is further segregated into 95 location areas of 2500 m$^2$ each. We are motivated to use a different network here rather than the previous network because it has too few location areas. Also, the objectives of the previous cases are to show the properties of $\gamma_h$, $\alpha$ and that HMTT works for single targets, whereas the next two cases (4 and 5) run on a network with more clusters, location areas, and multiple targets to illustrate the scalability of HMTT in terms of multiple targets and network size.

The number of targets in the region is kept to a maximum of 10 with the same dynamics as the previous cases. Fig. 8 shows the maximum prediction accuracy and minimum energy consumed over 3000 simulation runs, each lasting 8000 s. The convex hulls of each simulation run with respect to $\alpha$ is shown in Fig. 8(c). The effect of the balancing parameter $\alpha$ can again be seen clearly where a lower $\alpha$ gives more accurate tracking but consumes more energy and vice versa. The convex hulls lie in between the two extremes in an orderly manner, showing that
Fig. 8. Simulation results when $\gamma_h = 0.5$ for case 4) (multiple targets). (a) Maximum accuracy. (b) Minimum energy consumed. (c) Effect of balancing parameter $\alpha$.

$\alpha$ is an effective control parameter to balance between the two conflicting goals.

In the last set of results, we ran simulations of length 5000 s, each on a larger region of $1 \times 1$ km but with the same number of clusters and location areas. The number of targets remain the same, but their velocity range is increased to 40 m/s and variance tripled. Thus, we would expect accuracy to suffer. However, results from Fig. 9(a) show otherwise—accuracy could reach 0.872. This is probably due to an increase in size of a location area which leads to the perception that targets are actually much “slower” in transiting from area to area. Even though the variance is increased, LLA could handle the amount of uncertainty through learning. For the PES scheme, since straight-line trajectories are assumed, accuracy could only reach 0.776 for a 10% duty cycle. However, the energy consumed is 339 J and is about 32% more than that of HMTT. To reach an accuracy that is comparable to HMTT (0.840), the duty cycle of PES has to be increased to 50%, but that consumes 200% more energy at 770 J. On the other hand, HLPES cannot outperform HMTT in terms of energy consumption because the underlying predictor is inaccurate, leading HLA to increase its duty cycle to compensate for such inaccuracy. Hence, no clear trend can be seen in Fig. 9(b). This result may contradict that of case 3) on first glance. The reason why HLPES performed worse for the multiple target case is because HMTT gets more accurate with multiple targets as it gets more learning instances. With more accurate predictions, HMTT is able to increase its sleeping interval $\Delta t$ for better energy conservation.

VII. CONCLUSION AND FUTURE WORK

In this paper, we presented an effective distributed target-tracking algorithm called HMTT that can achieve higher energy savings than other known methods without compromising tracking accuracy. The proposed HMTT algorithm does not require strict and accurate localization methods, which is advantageous in wireless sensor networks where localization is inherently difficult. We introduced a mobility and tracking model to enable cluster heads to learn and adapt to target trajectories through a two-level HMDP. Hence, through learned HMDP policies, cluster heads are able to “memorize” target trajectory characteristics, which are vital to target tracking. We showed that our algorithm converges and derived its theoretical performance bound. Our future work will involve factoring in communication delays and sensor inaccuracies, as well as investigating the performance of HMTT against various sensing and communication ranges.
REFERENCES


Wai-Leong Yeow (S’04) received the B.Eng. degree (first-class honors) in computer engineering from National University of Singapore (NUS), Kent Ridge, Singapore, in July 2003, where he has been working toward the Ph.D. degree under the Agency for Science, Technology, and Research (A*STAR) Graduate Scholarship since August 2003. His research interests include communication issues in wireless sensor networks, especially their applications in target tracking, and distributed decision-making algorithms.

Chen-Khong Tham (S’92–M’95) received the M.A. and Ph.D. degrees in electrical and information sciences engineering, both from University of Cambridge, Cambridge, U.K.

He currently an Associate Professor with the Department of Electrical and Computer Engineering (ECE), National University of Singapore (NUS), Kent Ridge, Singapore. He holds a joint appointment with the Institute for Infocom Research (I²R), Singapore. His research interests are in coordinated quality-of-service management in wired and wireless computer networks and distributed systems, and distributed algorithms for resource allocation, machine learning, and decision making. He is the Supervisor of the Computer Networks and Distributed Systems Laboratory with the Department of Electrical and Computer Engineering, NUS, and was the Manager for Advanced Applications with the Singapore Advanced Research and Education Network (SingAREN) from 1997 to 1999. He held a 2004/05 Edward Clarence Dyason Universitas 21 Fellowship with the University of Melbourne, Parkville, Australia.

Wai-Chong (Lawrence) Wong (S’76–M’77–SM’93) received the B.Sc. (Hons.) and Ph.D. degrees in electronic and electrical engineering from Loughborough University, Loughborough, U.K.

He is currently with the Department of Electrical and Computer Engineering, National University of Singapore (NUS), Singapore. He was previously the Executive Director of the Institute for Infocomm Research (I²R) from November 2002 to November 2006. Since joining NUS in 1983, he has served in various positions at the department, faculty, and university levels, including Director of the Computer Centre, NUS, from July 2000 to November 2002 and Director of the Centre for Instructional Technology, NUS, from January 1998 to June 2000. Prior to joining NUS in 1983, he was a Member of the Technical Staff at AT&T Bell Laboratories, Crawford Hill Laboratory, NJ, from 1980 to 1983. He has published more than 200 journal and conference articles on mobile radio communication systems, wireless network protocols, robust digital speech communications, robust video communications systems, and multimedia network protocols and systems. He is the holder of four patents.

Dr. Wong was a recipient of the IEE Marconi Premium Award (1989); the IEEE Millennium Award (2000) (a professional service recognition award); the e-nnovator Awards 2000, Open Category (an innovation award that launched the online e-learning infrastructure at NUS); and the Best Paper Award at the IEEE ICME 2006 conference. He has also received several NUS teaching awards and commendations.