DCTC: Dynamic Convoy Tree-Based Collaboration for Target Tracking in Sensor Networks

Wensheng Zhang, *Student Member, IEEE* and Guohong Cao, *Associate Member, IEEE*

***Abstract*—Most existing work on sensor networks concentrates on finding efficient ways to forward data from the information source to the data centers, and not much work has been done on collecting local data and generating the data report. This paper studies this issue by proposing techniques to detect and track a mobile target. We introduce the concept of dynamic convoy tree-based collaboration, and formalize it as a multiple objective optimization problem which needs to find a convoy tree sequence with high tree coverage and low energy consumption. We propose an optimal solution which achieves 100% coverage and minimizes the energy consumption under certain ideal situations. Considering the real constraints of a sensor network, we propose several practical implementations: the conservative scheme and the prediction-based scheme for tree expansion and pruning; the sequential and the localized reconfiguration schemes for tree reconfiguration. Extensive experiments are conducted to compare the practical implementations and the optimal solution. The results show that the prediction-based scheme outperforms the conservative scheme and it can achieve similar coverage and energy consumption to the optimal solution. The experiments also show that the localized reconfiguration scheme outperforms the sequential reconfiguration scheme when the node density is high, and the trend is reversed when the node density is low.**

***Index Terms*—Convoy tree, prediction, reconfiguration, sensor networks, target tracking.**

I. Introduction

RECENT advances in digital electronics, microprocessor, microelectromechanics, and wireless communication have enabled the deployment of large scale sensor networks where thousands of small sensors are distributed over a vast field to obtain fine-grained, high-precision sensing data. Due to many attractive characteristics of sensor nodes such as small size and low cost, sensor networks [2], [12], [15], [19] become adopted to many military and civil applications, such as target tracking, surveillance, environmental control, and health care.

This paper studies the problem of detecting and tracking a mobile target, and monitoring a particular region surrounding the target in sensor networks. As shown in Fig. 1, the sensor nodes surrounding an adversary tank detect and track the tank and its surrounding area which may include enemy soldiers. These nodes collaborate among themselves to aggregate data about the tank as well as its surrounding area, and one of them (i.e., the root) generates a data report. The data report can be saved locally waiting for other node's query, or can be forwarded to single or multiple data centers (the sinks), which can be a static command center or moving soldiers. Other examples can be found in applications such as tracking an important target (e.g., an important person) in a parade, or a valuable animal (e.g., a panda) in the forest. As design goals, the sensor nodes surrounding the moving target should promptly provide robust and reliable status information about the mobile target and the region around it in an energy efficient way, and the network should forward this information to the sinks in a fast and energy efficient way.

Most existing researches in sensor networks, e.g., the directed diffusion [10], [13], LEACH [9], and two-tier data dissemination (TTDD) [22], concentrate on finding efficient ways to forward the data report to the data center, and not much work has been done on how to detect the mobile target and generate robust and reliable reports in an energy efficient way. Recently, Chu *et al.* [6], [23] studied the problem of tracking a mobile target using an information-driven approach. However, their approach assumed that a single node close to a target can detect the status of the target, and did not consider the collaboration among nodes that can detect the target at the same time. Since sensor nodes deployed in current sensor networks do not have a large sensing distance, or a high level of sensing accuracy and node reliability, Cerpa *et al.* [5] suggested that multiple nodes surrounding the target should collaborate to make the collected information more complete, reliable, and accurate. However, no concrete algorithm was given.

In this paper, we propose a dynamic convoy tree-based collaboration (DCTC) framework to detect and track the mobile target and monitor its surrounding area. DCTC relies on a tree structure called *convoy tree*,[1] which includes sensor nodes around the moving target, and the tree is dynamically configured to add some nodes and prune some nodes as the target moves. Fig. 1 illustrates how to use the convoy tree to track a mobile target. As the target first enters the detection region, sensor nodes that can detect the target collaborate with each other to select a root and construct an initial convoy tree. Relying on the convoy tree, the root collects information from the sensor nodes and refine these information to obtain more complete and accurate information about the target using some classification algorithms [11], [12]. As the target moves, some nodes in the tree become far away from the target and are pruned. Since most sensor nodes stay asleep before the target arrives for power saving, the root should predict the target moving direction and activate the right group

Manuscript received October 21, 2002; revised February 16, 2003, May 22, 2003; accepted June 9, 2003. The editor coordinating the review of this paper and approving it for publication is M. Zorzi. This work was supported in part by the National Science Foundation under Grant CAREER CCR-0092770 and Grant ITR-0219711.

The authors are with the Department of Computer Science and Engineering, The Pennsylvania State University, University Park, PA 16802 USA (e-mail: wezhang@cse.psu.edu; gcao@cse.psu.edu).

Digital Object Identifier 10.1109/TWC.2004.833443

[1]We use the word "convoy tree" to differentiate it from the normal tree structure since convoy tree is a moving tree which tracks the target.

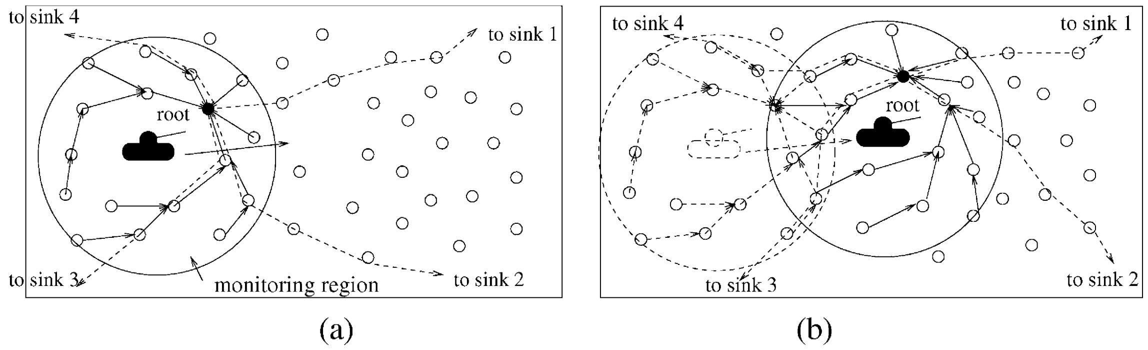

Fig. 1. Using convoy tree to track the target. (a) Data collection. (b) Tree reconfiguration.

of sensor nodes so that these nodes can detect the target as soon as the target shows up. As the convoy tree reconfigures itself, the root may also need to be changed to optimize the communication overhead. Fig. 1(b) shows how the convoy tree reconfigures itself with a new root.

A big challenge of implementing the DCTC framework is how to reconfigure the convoy tree in an energy efficient way as the target moves. To address this problem, we first formalize it as an optimization problem of finding a min-cost convoy tree sequence with high tree coverage, and give an optimal solution (o-DCTC) based on dynamic programming. Considering the constraints of sensor networks, we propose some practical solutions. Specifically, we propose two tree expansion and pruning schemes: the *conservative* scheme and the *prediction-based* scheme; and two tree reconfiguration schemes: the *sequential reconfiguration* and the *localized reconfiguration*. We also evaluate the performance of the optimal solution and the practical implementations through extensive simulations. Based on the simulation results, when the same reconfiguration scheme is used, the prediction-based scheme outperforms the conservative scheme and it can achieve a similar coverage and energy consumption to the optimal solution. When using the same scheme for tree expansion and pruning, the localized reconfiguration scheme outperforms the sequential reconfiguration scheme when the node density is high, and the trend is reversed when the node density is low.

The remainder of the paper is organized as follows. Section II describes the system model and the basic framework of DCTC. Section III presents an optimal DCTC based on dynamic programming. In Section IV, we propose practical solutions to improve the performance considering the constraints of sensor networks. Section V evaluates different solutions and compares them to the optimal solution. Section VI concludes the paper.

II. System Model

A sensor network can be modeled as a graph $G(V, E)$, where each vertex represents a sensor node, and there is an edge between two nodes when they are within each other's communication range. Each node is aware of its own location by using the global positioning system (GPS) [1] or other techniques such as triangulation [4]. Each sensor node can be in the active mode or sleep mode. In the active mode, the node can sense the target, send, and receive data. An active node can select its transmission power from the following n levels: $u_1 \leq u_2 \leq \cdots \leq u_n$.

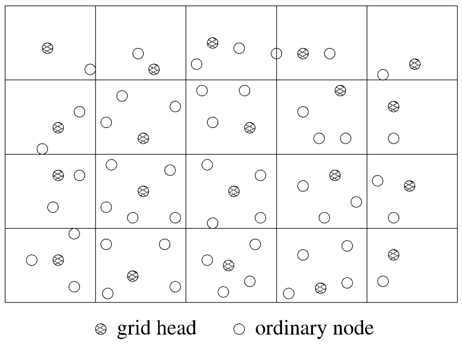

Fig. 2. Dividing a sensor network into grids.

Corresponding to the power level it selects, it has one of the following communication ranges: $d_1 \leq d_2 \leq \cdots \leq d_n$. Here, the relation between the power level and the communication range satisfies the following energy model [8]: $u_i = a{*}d_i^2 + c$, where a and c are constants.

To save power, the sensor nodes stay in the sleep mode most of the time based on the geographical adaptive fidelity (GAF) protocol [20]. As shown in Fig. 2, the sensor network is divided into grids, where each pair of nodes in neighboring grids can communicate directly with each other. When there is no target close to a grid, only the grid head is required to be awake, and other nodes only need to wake up periodically. Each node maintains some information about the nodes in the same grid, e.g., the locations. Through the discovery process of the GAF protocol, each node can learn the information about other nodes in the same grid. When a mobile target[2] enters the detection region, the grid head will wake up all nodes inside the grid.

Let t_s (t_e) denote the time when the target enters (leaves) the detection region of the network. At any time t $(t_s \leq t < t_e)$, the set of sensor nodes that should participate in detecting the target is denoted as S_t. Note that, S_t is application specific, and may be affected by several factors such as the sensing capacity of sensor nodes, the precision and reliability demands on the sensing results, the size of the target, etc. In this paper, we assume that S_t includes all the nodes whose distance to the current location of the target is less than d_s. Also, we call the circle that is centered at the current location of the target and has a radius of d_s as the monitoring region of the target at time t.

[2]In this paper, we only consider the problem of detecting a single mobile target at one time.

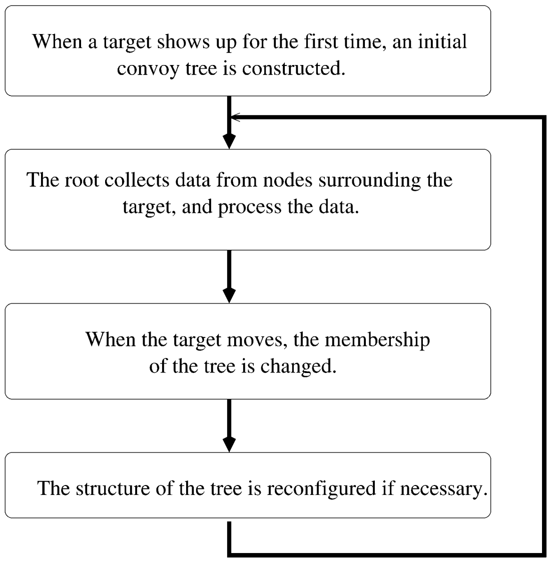
When a target shows up for the first time, an initial convoy tree is constructed.
The root collects data from nodes surrounding the target, and process the data.
When the target moves, the membership of the tree is changed.
The structure of the tree is reconfigured if necessary.

Fig. 3. Basic structure of DCTC.

Fig. 3 illustrates the basic design of DCTC. With DCTC, a convoy tree of a target is formed as soon as the target enters the detection region of the network. Every certain time interval[3], each node in the tree generates a sensing report and sends the report to its parent, which forward the report to the root. After the root has received all the reports, it processes them to generate a final report that will be sent to the sinks. From the processing results, the root obtains the current location of the target. Based on this information, it may add some new nodes, prune some existing nodes, or reconfigure itself. The convoy tree at time t is denoted as $T_t(R)$, which is formally defined as follows.

Definition 1 (Convoy Tree): Convoy tree $T_t(R) = \langle V_t, E_t \rangle$, where V_t is the set of nodes that are awake and participate in detecting the target at time t, E_t is the set of edges connecting nodes to their parents, and R is the root of the tree.

One important attribute of a tree T_t is the tree coverage, which is defined as

$$\frac{\mid V_t \cap S_t \mid}{\mid S_t \mid}.$$

Intuitively, the tree coverage is the number of nodes in the tree as a fraction of the number of nodes that can detect the target. We use $V_t \cap S_t$ instead of V_t, because some nodes in the tree cannot detect the target but have not been pruned yet. When all nodes that can detect the target are in the tree, the tree coverage reaches 100%. Increasing the tree coverage can increase the number of nodes involved in the target detection, which helps obtain more accurate information about the target. Some nodes may only detect part of the target, or some angle of the target. Some sensor nodes may be malicious or have malfunction. In such an environment, redundancy is helpful. With some level of redundancy, the root can obtain more accurate information about the target. Note that the root will remove some redundant information and only send the essential information to the data center. Certainly, the root may fail or even be a malicious node. To deal with this kind of problem, other nodes need to monitor what happens with the root. When the root fails, its neighboring nodes need to collaborate to elect a new one. They may even periodically send some encrypted information to the data center directly if the root is suspected to be malicious. Since this is not the major concern of the paper, we will not further discuss it.

[3]To make the presentation easy, we assume that time is discrete and the time interval is 1.

The other important attribute of tree $T_t(R)$ is the energy consumed for collecting the sensing data via this tree. Let r denote the size of the sensing data generated by node i, $e(i, R)$ denote the total energy consumed by transmitting one unit of information from node i to R through the path on the tree. The amount of energy consumed for data collection is defined as

$$E^d(T_t(R)) = r * \sum_{i \in V_t} e(i, R).$$

The convoy tree reconfigures itself as the target moves. A collection of trees at different time interval form a convoy tree sequence, which is denoted as

$$\Gamma(t_s, t_e, T_s(R)) = \langle T_s(R_s), T_{t_s+1}(R_{t_s+1}), \ldots, T_{t_e}(R_{t_e}) \rangle$$

where t_s is the start time of the first data collection process. When the parameter $T_s(R)$ is not necessary, it will be removed for simplicity.

The average tree coverage of $\Gamma(t_s, t_e)$ is defined as

$$\frac{\sum_{t=t_s}^{t_e} \frac{|V_t \cap S_t|}{|S_t|}}{t_e - t_s + 1}.$$

The total energy consumption of $\Gamma(t_s, t_e, T_s)$ is defined as

$$E(\Gamma(t_s, t_e, T_s)) = E^d(T_s) + \sum_{t=t_s+1}^{t=t_e} [E^t(T_{t-1}, T_t) + E^d(T_t)]$$

where $E^t(T_{t-1}, T_t)$ is the energy consumed by the transformation from T_{t-1} to T_t. A convoy tree sequence $\Gamma(t_s, t_e, T_s)$ is called a min-cost convoy tree sequence if $[\forall \Gamma'(t_s, t_e, T_s))(E(\Gamma(t_s, t_e, T_s)) \leq E(\Gamma'(t_s, t_e, T_s))]$.

Generally speaking, there is a tradeoff between tree coverage and energy consumption, and we should not optimize only one of them without considering the constraint of the other. This can be further explained by an example. If there is no constraint on the energy consumption, the tree coverage can reach 100% by adding all nodes to the tree. Similarly, without considering the constraints of the tree coverage, finding a min-cost convoy tree sequence is trivial. However, it is a challenge to find a min-cost convoy tree sequence whose average coverage is larger than a certain value. In the next section, we study a special case of this problem: finding a min-cost convoy tree sequence that can reach 100% coverage, which can only be solvable under ideal situations.

III. o-DCTC Scheme

If each node has the information about the network topology, and the moving trace of the target is known, we can design an o-DCTC to find a min-cost tree with the constraint that the tree

```
Notations:

  • $\Phi_t$: a set of convoy tree sequence.

The o-$DCTC$ algorithm:
Function $DCTC(t, t_e, \Phi_t)$
1    if $t = t_e$ then
2        find $\Gamma \in \Phi_t$, such that, $\forall \Gamma' \in \Phi_i (E(\Gamma) \le E(\Gamma'))$;
3        return $E(\Gamma)$;
4    else
5        $\Phi_{t+1} = \emptyset$;
6        for each $T_{t+1}$ which is a min-convoy tree at time $t+1$
7            $min\_eng = \infty$;
8            for each $\Gamma_t \in \Phi_t$
9                append $T_{t+1}$ to $\Gamma_t$ to get $\Gamma_{t+1}$;
10               if $E(\Gamma_{t+1}) \le min\_eng$ then
11                   $min\_eng = E(\Gamma_{t+1})$; $\Gamma' = \Gamma_{t+1}$;
12           $\Phi_{t+1} = \Phi_{t+1} + \{\Gamma'\}$;
13       $DCTC(t+1, t_e, \Phi_{t+1})$
begin
    $DCTC(t_s, t_e, \{\langle T_s \rangle\})$
end.
```

Fig. 4. Optimal DCTC scheme.

coverage is 100%. Since the whole target trace is known, the tree coverage can reach 100% by expanding the convoy tree to include all nodes and only those nodes that are included in the monitoring region of the target. Also, each of these nodes can be pruned from the tree as long as they are out of the monitoring region.

Different from the general problem of finding a min-cost convoy tree sequence, in this section, we consider convoy trees that contain and only contain nodes that are within the monitoring region of a target. To help understand o-DCTC, we provide the formal definitions for this special kind of trees and the min-cost convoy tree sequences that contain only this kind of trees.

Definition 2 (Min-Convoy Tree): A convoy tree $T_t(R)$ is a min-convoy tree if and only if $T_t(R)$ contains and only contains nodes in S_t.

Definition 3 (Min-Convoy Tree Sequence): A convoy tree sequence $\Gamma(t_s, t_e)$ is a min-convoy tree sequence if and only if $\forall \Gamma'(t_s, t_e)(E(\Gamma(t_s, t_e) \le E(\Gamma'(t_s, t_e) \wedge$ trees in $\Gamma(t_s, t_e)$ and $\Gamma'(t_s, t_e)$ are all min-convoy trees).

In the rest of the section, we present a centralized algorithm that uses dynamic programming to compute a min-convoy tree sequence in Fig. 4, and prove some properties of this algorithm. The algorithm is based on (1), located at the bottom of the page.

Here, $\mathcal{T}_t$ is the set of all min-convoy trees at time t. We show the correctness of the o-DCTC algorithm by proving that o-$DCTC(t, t_e, T_t)$ in (1) can compute a min-convoy tree sequence starting from T_t.

Theorem 1: Let $\Gamma_m(t, t_e, T_t)$ be a min-convoy tree sequence. o-$DCTC(t, t_e, T_t) = E(\Gamma_m(t, t_e, T_t))$, where o-DCTC is defined in (1).

Proof: By induction on t.

1) **Base:** When $t = t_e$, $E(\Gamma_m(t, t_e, T_t)) = E(\langle T_t \rangle) = E^d(T_t)$, and o-$DCTC(t, t_e, T_t) = E^d(T_t)$. Therefore, $E(\Gamma_m(t, t_e, T_t)) =$ o-$DCTC(t, t_e, T_t)$.
2) **Induction:** Assume that o-$DCTC(t, t_e, T_t) = E(\Gamma_m(t, t_e, T_t))$ for $t > t_e - n$, $n \ge 1$. We want to prove that it holds when $t = t_e - n$.
    a) Let $\Gamma_m(t, t_e, T_t) = \langle T_t, T'_{t+1}, \ldots, T'_{t_e} \rangle$. Thus,

$$\begin{aligned}
&E(\Gamma_m(t, t_e, T_t))\\
&= E^d(T_t) + E^t(T_t, T'_{t+1}) + E(\langle T'_{t+1}, \ldots, T'_{t_e} \rangle)\\
&\ge E^d(T_t) + E^t(T_t, T'_{t+1}) + E(\Gamma_m(t+1, t_e, T'_{t+1}))\\
&= E^d(T_t) + E^t(T_t, T'_{t+1}) + o\text{-}DCTC(t+1, t_e, T'_{t+1})\\
&\ge o\text{-}DCTC(t, t_e, T_t).
\end{aligned}$$

    b) On the other hand, let

$$\begin{aligned}
&o\text{-}DCTC(t, t_e, T_t)\\
&\quad = E^d(T_t) + E^t(T_t, T''_{t+1}) + o\text{-}DCTC(t+1, t_e, T''_{t+1}).
\end{aligned}$$

Thus,

$$\begin{aligned}
&o\text{-}DCTC(t, t_e, T_t)\\
&\quad = E^d(T_t) + E^t(T_t, T''_{t+1}) + E(\Gamma_m(t+1, t_e, T''_{t+1}))\\
&\quad = E(\langle T_t, T''_{t+1}, \ldots, T''_{t_e} \rangle) \ge E(\Gamma_m(t, t_e, T_t)).
\end{aligned}$$

Therefore, $\Gamma_m(t, t_e, T_t) =$ o-$DCTC(t, t_e, T_t)$.

Theorem 2: If the maximum number of min-convoy trees at any time is M, the computational complexity of the o-DCTC algorithm is $O((t_e - t_s + 1) * M^2)$.

Proof: This can be drawn by examining the function *DCTC*, which is executed for $(t_e - t_s) + 1$ times. In each execution, the maximum number of executing the for loop of lines 6–12 is equal to M, and the maximum number of executing the for loop of lines 8–11 is also equal to M. Thus, the computational complexity is equal to $O((t_e - t_s + 1) * M^2)$ in the worst case.

Tree Expansion/Pruning and Reconfiguration in o-DCTC: Until now, we have been discussing the o-DCTC scheme in an abstract way, and have not mentioned how to calculate the energy consumed for transforming one tree to another. To do this, we need to know how nodes are added/pruned and how a tree is reconfigured in the ideal situation. Note that the method for calculating the energy consumption for data collection is already defined in the previous section.

First, we introduce an algorithm used to calculate a min-cost convoy tree. It is similar to the *Dijkstra' algorithm* for solving the single-path problem in a weighted graph [7]. The correctness

$$o\text{-}DCTC(t, t_e, T) = \begin{cases} E^d(T), & \text{if } t = t_e;\\ MIN_{T_{t+1} \in \mathcal{T}_{t+1}}\{E^d(T_t) + E^t(T_t, T_{t+1}) + o\text{-}DCTC(t+1, t_e, T_{t+1})\}, & \text{if } t < t_e \end{cases} \qquad (1)$$

proof is also similar to that provided in [7]. Note that all the notations used in this algorithm have been defined before.

```
Algorithm 1 $MinCostTree(V, e, R)$
1 $Q = V$
2 While $Q \neq \emptyset$
3   find $u \in Q$ such that $(\forall v \in Q)(e(u,R) \leq e(v,R))$;
4   for each $v$ that is a neighbor of $u$
5     if $e(v,u) + e(u,R) < e(v,R)$ then
6       $e(v,R) = e(v,u) + e(u,R)$; change $v$'s parent to be $u$.
7   $Q = Q - \{u\}$.
```

At time $t = t_s, t_s + 1, \ldots, t_e$, the convoy tree $T_t(R)$ needs to expand itself to include nodes that will detect the target at time $t + 1$. The tree expansion procedure is defined as follows:

1) R defines the new set of nodes to join the tree as $S_{\text{new}} = S_{t+1} - V_t$;
2) R multicasts a message $wake\_up(L_{t+1})$ to the heads of the grids that contain nodes in S_{new}; receiving the *wake_up* , the grid head wakes up those nodes;
3) after a node wakes up, it calls $MinCostTree(S_{t+1}, e, R)$ to compute the min-cost convoy tree and joins the tree by attaching to the parent node computed by the algorithm.

The tree pruning process is similar to the tree expansion process. Here, the notification message is called power_down, and only nodes in $S_{old} = V_t - S_{t+1}$ are notified. When a node receives the power_down message, it leaves the tree and returns to sleep if it is not the grid head.

A tree reconfiguration procedure which transforms $T_{t+1}(R)$ to $T_{t+1}(R')$, where $R' \in V_{t+1}$, is defined as follows:

1) R' multicasts a message $reconf(R', L_{t+1})$ to the heads of the grids that contain the nodes in S_{t+1}; receiving the message, the grid head broadcasts the message to those nodes;
2) after a node receives $reconf(R', L_{t+1})$, it calls $MinCostTree(S_{t+1}, e, R')$ to compute the min-cost convoy tree and changes its parent to the one computed by the algorithm.

From the description of these processes, we know that the overhead for reconfiguring a tree [i.e., $E^t(T_t(R), T_{t+1}(R'))$] includes the energy consumed for transmitting the messages *wake_up*, *power_down*, and *reconf* during the reconfiguration.

IV. Design and Implementation of DCTC

Although the o-DCTC scheme can optimize the energy consumption and achieve a 100% tree coverage, it is not practical since it assumes that the moving target trace is known as *a priori* and each node has knowledge about the network topology. In this section, we propose some practical solutions to implement the DCTC framework.

A. Constructing the Initial Tree

When a target first enters the detection region of a sensor network, the sensor nodes that are awake and close to the target can detect it. Then, these sensor nodes need to construct an initial convoy tree by first selecting a node to be the root of the tree. Many leader election algorithms [18] can be used to select the root. Since this is not the major concern of this paper, we only present the basic idea of a simple solution, which selects the node closest to the target as the root. If two or more nodes have the same distance to the target, node id (can be the MAC address) is used to break the tie. This selection criterion is based on the following intuitive reasons: First, if the root is close to the target, i.e., close to the geographic center of the nodes in the tree, the tree should have a short height and small energy consumption during data collection. Second, since there may be multiple moving sinks [22] distributed in the network, selecting a root that is far away from the sensing nodes may not be a good solution. Note that this is different from some existing work [10] where collected data will be forwarded to the same sink. In some cases, the new root may be some special powerful node, which can cache the data and wait for other nodes to query the data. In this case, the root selection process will be much simpler.

This algorithm has two phases. In the first phase, each node i broadcasts to its neighbors an $election(d_i, id_i)$ message with its distance to the target (d_i) and its own id. If a node does not receive any election message with (d_j, id_j) that is smaller than (d_i, id_i), it becomes a root candidate. Otherwise, it gives up and selects the neighbor with the smallest (d_j, id_j) to be its parent. Since the nodes may not be within one-hop communication range of each other,[4] two or more root candidates may exist after the first phase. In the second phase, each root candidate i floods a $winner(d_i, id_i)$ message to other nodes that can detect the target. When a root candidate i receives a winner message with (d_j, id_j) that is smaller than (d_i, id_i), it gives up the candidacy and constructs a path to node j by reversing the path on which the winner message is sent. Eventually, all root candidates give up but one with the smallest (d_i, id_i) becomes the root. These election and winner messages are limited to only nodes that can detect the target. The selection process only takes a very short time, so there should not be a case where new nodes keep adding into the root election process. The initial convoy tree constructed in this way may not be a min-cost convoy tree, and may have low coverage. The tree expansion and reconfiguration schemes, which will be presented later, can be used to reconstruct a better convoy tree with high coverage and low cost.

B. Tree Expansion and Pruning

For each time interval, the root adds some nodes and removes some nodes as the target moves. To identify which node to be added and removed, we study two schemes: the *conservative* scheme and the *prediction-based* scheme.

In the conservative scheme, as shown in Fig. 5(a), the nodes that should be in the tree are those whose distance to the current location of the target is less than $v_t + \beta + d_s$, where v_t is the current velocity of the target, β is a parameter, and d_s is the radius of the monitoring region. This scheme assumes that the target may move in any direction with a velocity less than $v_t + \beta$. Obviously, β should be large enough in order to achieve a high tree coverage. However, if β is too large, the expanded tree

[4]We assume there is no network partition.

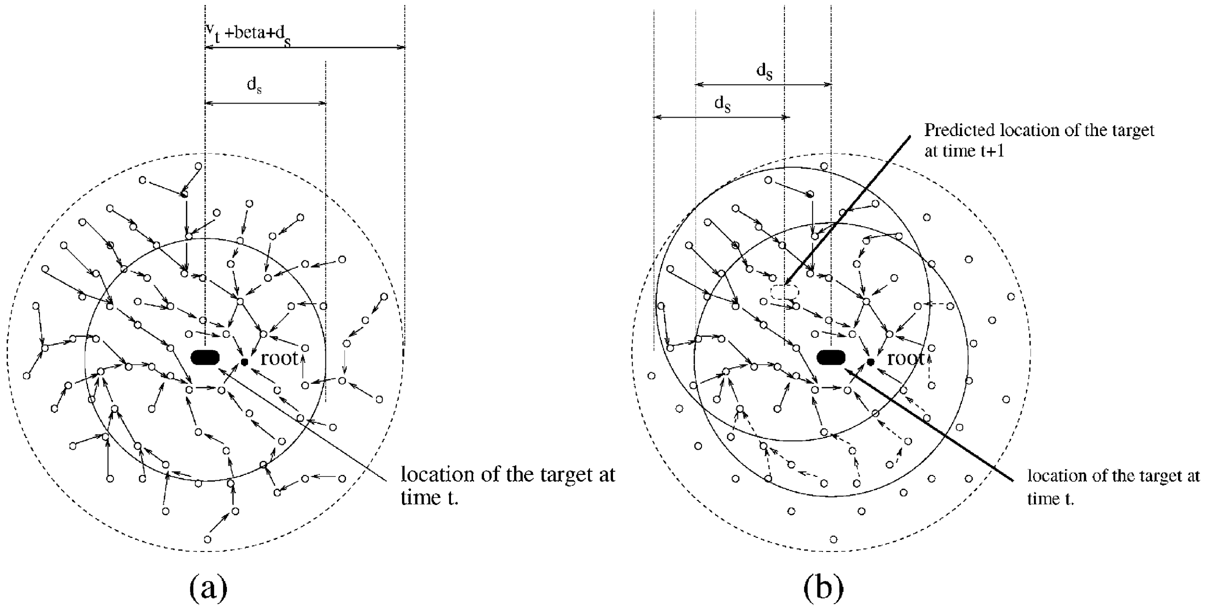
v_t +beta+d_s
d_s
root
location of the target at time t.
(a)
d_s
d_s
Predicted location of the target at time t+1
root
location of the target at time t.
(b)

Fig. 5. Tree expansion and pruning. (a) Conservative scheme. (b) Prediction-based scheme.

may contain a lot of redundancy; i.e., many nodes which should not be in the tree are added to the tree. This will increase the power consumption since these nodes cannot be in the power save mode.

The prediction-based scheme is proposed to reduce the redundant nodes added by the conservative scheme. This scheme assumes that certain movement prediction techniques can be used to predict the future location of the target. Therefore, only nodes located within the estimated monitoring region are added to the tree. As shown in Fig. 5(b), only nodes whose distance to the predicted location are less than d_s are added. Since the prediction may not be accurate, the tree coverage may be reduced. However, considering the target moving direction is not frequently changing, the chance of correctly predicting the target moving direction is high. Hence, the advantage of using prediction outweighs its negative effects. Due to space limit, we do not present the details of the movement prediction technique, which can be found in [3] and [14].

After the nodes that should be added to the tree and those that should be pruned from the tree are identified, the root notifies them by sending notification messages to the heads of the grids that contain these nodes. A grid head receiving a prune message immediately broadcasts it to the nodes in the grid, and the nodes receiving the message leave the tree and return to sleep. When a grid head receives a join message, it wakes up the sensor nodes in its grid. It also requests for the locations and costs of the nodes within its neighboring grids that are closer to the root. After receiving all these information, they are broadcast to the nodes in the grid, which find the appropriate parents to minimize their costs of sending packets to the root. The algorithm for finding the parent is the same as that used in the tree reconfiguration schemes, which will be presented next.

C. Tree Reconfiguration

1) Basic Scheme: When the target moves, the sensor nodes that participate in the detection change accordingly. If many nodes become far away from the root, a large amount of energy may be wasted for them to send their sensing data to the root. To reduce the energy consumption, the root should be replaced by a node closer to the center of the monitoring region (i.e., the moving target), and the tree should be reconfigured. Due to the reconfiguration cost, the tree should not be reconfigured too frequently, especially when the current root is still close to the target. In our solution, the reconfiguration is triggered by a simple heuristic, which is based on the distance between the root and the current location of the target. If this distance is larger than a threshold, the tree should be reconfigured; otherwise, it should not. We set the reconfiguration threshold to be

$$d_m + \alpha * v_t$$

where d_m is a parameter to specify the minimum distance that triggers the reconfiguration, v_t is the velocity of the target at time t, and α is a parameter to specify the impact of the velocity.

If the tree should be reconfigured based on the heuristic, a new root should be selected. This new root should be a node close to the current location of the target. When multiple nodes exist, one of them can be randomly chosen. We already explained part of the reason in Section IV.A. Choosing a node close to the target is helpful for constructing a tree with short height and less energy consumption during data collection. This will be a good option when multiple sinks exist, or when the data has to be cached in the network, waiting for other nodes to query the data. The process for root migration is briefly described as follows after the new root has been chosen. The current root first finds the grid that covers the current location of the target. Then, it sends a migration request to the grid head. Receiving the request, the grid head forward the message to the new root.

After the root has been changed, the tree needs to be reconfigured to minimize the energy consumption for data collection.

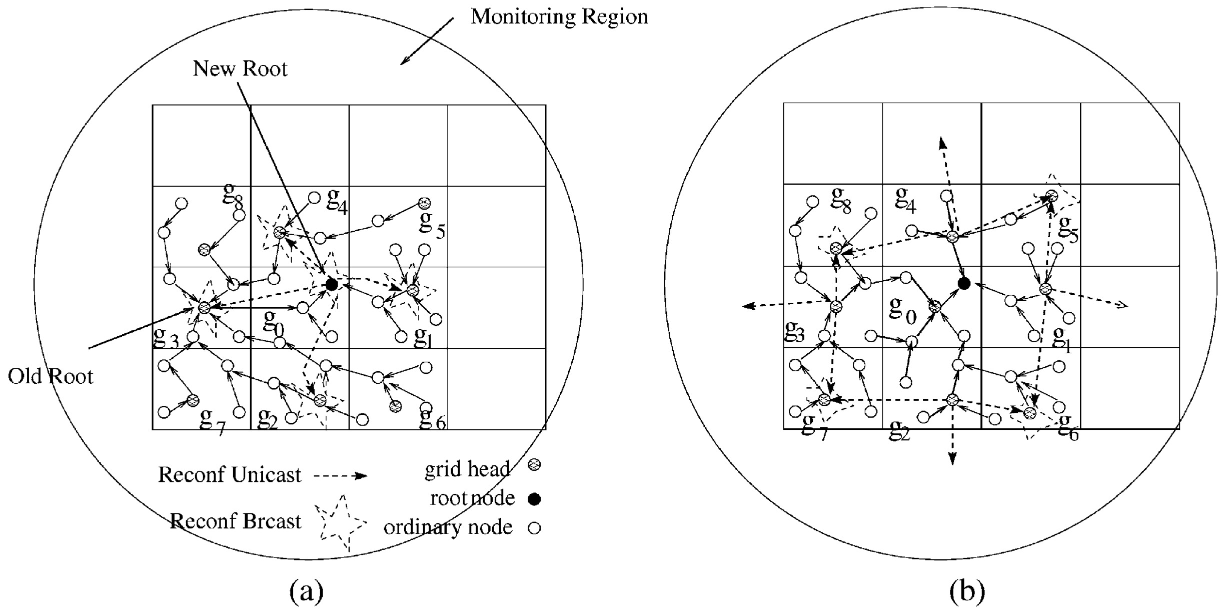

Fig. 6. Sequential tree reconfiguration. (a) Step 1. (b) Step 2.

In Section III, we presented the *MinCostTree* algorithm to construct a min-cost convoy tree under the assumption that each node is aware of the locations of all nodes in the tree. Due to the high overhead of maintaining these information, this algorithm may not be suitable for a real sensor network.

Another option is based on broadcasting. In this approach, the new root first broadcasts a *reconf* message to its neighbors. A node receiving the *reconf* rebroadcasts the message and attaches its location and the cost of sending a packet to the root. When any other node receives the *reconf*, it waits for certain time interval and collects the *reconf* messages sent during this interval. After that, it changes its parent to be a selected neighbor, so that the cost of sending packets to the root via this neighbor is smaller than via any other known neighbors. After this, it broadcasts a *reconf* attaching its location and the cost of sending a packet to the root. This process will continue until all nodes that are within the monitor region are added to the tree. Using this approach, a node does not need to maintain the information about other nodes since the information can be obtained from neighbors during the reconfiguration process. However, the broadcasting overhead is too high, and it is difficult for a node to decide how long it should wait to collect the information from its neighbors. To address this issue, we present two practical tree reconfiguration schemes: the sequential reconfiguration scheme and the localized reconfiguration scheme.

2) Sequential Reconfiguration: The proposed sequential reconfiguration scheme is based on the grid structure. The main idea has been illustrated in Fig. 6. As shown in Fig. 6(a), the new root (in grid g_0) initiates the reconfiguration by broadcasting a *reconf* message to the nodes in the its own grid, and sending the message to the heads of its neighboring grids (i.e., g_1, g_2, g_3 and g_4). The *reconf* includes information about nodes in its own grid such as the locations of these nodes and the cost of sending data to the root. Having received the *reconf*, the grid head needs to do the following. First, it broadcasts the messages to the nodes in its own grid. Second, based on the information attached with these messages, it finds the parent for each node in the grid, so that the cost of sending packets from the node to the root is minimized. Finally, as shown in Fig. 6(b), it sends a *reconf* which includes the location and cost of each node in the grid, to the heads of its neighboring grids that are far away from the root, and these grid heads will repeat the same process until all nodes in the monitor region are added to the tree. Based on the property of the grid structure, the grid head of g_0 may not be within the communication range of the grid head of g_5, g_6, g_7, or g_8. Hence, it only sends *reconf* to the grid heads of g_1, g_2, g_3, g_4, and let them to further propagate the *reconf* requests. Also, each ordinary node needs to find its parent that can minimize its cost to send packets to the root based on the information gathered from the grid head. A formal description of this algorithm is shown in Fig. 7.

3) Localized Reconfiguration: The sequential reconfiguration algorithm has some drawbacks. During each reconfiguration process, large lists containing the information about all nodes in a grid are transmitted between grids, and are broadcast within grids. This may create significant amount of traffic, especially when the node density is very high or the reconfiguration is frequent. In this section, we propose a localized reconfiguration scheme which provides two optimizations to address this problem.

In the first optimization, we can remove the cost information from the reconfiguration message since a heuristic can be used to estimate the cost, as long as the distance between two nodes is known. Based on the results from [17], the cost of sending one unit of information from node i to node j can be estimated by $\sigma * v(d_{i,j})$, where $\sigma > 1$ is a system parameter used to estimate the relationship between the minimum cost and the real cost. A large σ means a high cost. In our simulations, σ is set to 2.0. $v(x)$ is a function defined as $c*(a/c)^{1/2}*x+a*(a/c)^{-1/2}*x$, where a and c are parameters related to the power model. It has

Notations:

- g_i: a grid. g_0 is the grid of the new root.
- l_i: a list recording the locations of the nodes in grid g_i and the cost for sending packets to the root. Formally, $l_i = \{\langle k, e(k,R)\rangle \mid k \text{ is a node in } g_i\}$.
- $reconf(R, l_i)$: a reconfiguration message sent by the head of g_i.
- $d_{i,j}$: the distance between node i and node j.

```
Procedure $MinEngPath(R, l_1, l_2)$
Sort items in $l_1$ in the increasing order of $d_{i,R}$, where $\langle i, e(i,R)\rangle$ is an item in $l_1$.
for each $\langle i, e(i,R)\rangle \in l_1$
    find $\langle j, e(j,R)\rangle \in l_2$, such that $(\forall\langle k, e(k,R)\rangle \in l_2)(e(i,j) + e(j,R) \le e(i,k) + e(k,R))$;
    node $i$'s parent is set to $j$, $e(i,R) = e(i,j) + e(j,R)$;
    $l_2 = l_2 + \{\langle i, e(i,R)\rangle\}$;
The Sequential Tree Reconfiguration Algorithm

(C.1) $R$ sends $reconf$ to nodes in its grid, $g_0$. Those nodes call $MinEngPath(R, l_0, \emptyset)$ to compute their
      costs and parents.
(C.2) $R$ sends $reconf(R, l_0)$ to the head of each neighboring grid.
(C.3) When the head of $g_i$ receives $reconf(R, l_j)$ and it has some member nodes within the monitor
      range:
          if $g_i$ has received $reconf$ from any neighboring grids that are closer to $R$ then
              call $MinEngPath(R, l_i, \bigcup l_j)$
              send $reconf(R, l_i)$ to the head of each neighboring grid that is far away from $R$.
```

Fig. 7. Sequential tree reconfiguration algorithm.

been proved that $v(d_{i,j})$ is the minimum cost for sending one unit of information from node i to node j [17]. To implement this idea, *MinEngPath* of Fig. 7 should be modified as follows:

```
MinEngPath$(R, i, l_2)$
(1) find $\langle j, e(j,R)\rangle \in l_2$, such that
$(\forall\langle k, e(k,R)\rangle \in l_2)\{e(i,j) + \sigma * v(d_{j,R}) \le e(i,k) + \sigma * v(d_{k,R})\}$
(2) node $i$ changes its parent to be $j$;
$e(i,R) = e(i,j) + \sigma * v(d_{j,R})$.
```

In the second optimization, the information about the node locations can be removed from the messages, if the receiver has got this information about the sender recently. This can be easily implemented by caching the location information about other nodes for some time, and certainly under the assumption that nodes do not move frequently.

Comparing to the sequential reconfiguration scheme, this scheme can greatly reduce the reconfiguration overhead. This overhead reduction can be significant when the node density is high or when the reconfiguration is frequent. However, the tree constructed by this scheme may have high data collection overhead in terms of energy when compared to the sequential reconfiguration, especially when the node density is low. Thus, there is tradeoff between the reconfiguration overhead and data collection overhead, and we will use simulations to evaluate the overall energy consumption of both schemes under different system settings.

V. Performance Evaluations

A. Simulation Model

We developed a simulator based on *ns2* (version 2.1b8a) [16], to evaluate and compare the performance of the o-DCTC scheme and the proposed practical solutions. In this simulator, the MAC protocol is based on IEEE 802.11, and the tworay ground propagation model is adopted. The transmission range of each node can be 10, 2, 30, and 40 m, and the transmission power used to support each of the transmission ranges is calculated using the model in [8]. Sensor nodes are distributed over a 500×500 m^2 flt field, which is divided into 17×17 m^2 grids. In each experiment, 5000 or 1500 sensor nodes are deployed to simulate a dense setting or a sparse setting.

To simulate the movement of a target, we use a mobility model similar to [21]. In this model, we consider the following characteristics of a moving target: First, the acceleration of the target is highly correlated, i.e., if it is accelerating at time t, it is likely to continue accelerating at time $t+1$. Second, the target may also have sudden and unexpected changes in acceleration. We use a *command acceleration* to generate the unexpected changes in acceleration, which is modeled as a Markov chain with a finite number of states, $C_0, C_1, \ldots, C_8$ ($C_i = i * 0.5 - 2.0$). The transition probability from C_i to C_j, denoted as $\theta_{i,j}$, is 0.5 if $i = j$ and 0.0625 if $i \neq j$. We also use a *random acceleration* to model the gap between adjacent and highly correlated acceleration states, and it is uniformly distributed between -1.0 and 1.0. At the beginning of the simulation, we assume that the target shows up at the center of the field with an arbitrary initial moving direction and velocity. When evaluating the prediction-based scheme, the movement of the target can be predicted with an accuracy of p, varying from 0.6 to 0.9. A mis-prediction happens when the acceleration of the target is changed, but predicted to be unchanged. Table I lists most of the simulation parameters.

B. Simulation Results

In this section, we compare different tree expansion, pruning, and reconfiguration schemes in terms of tree coverage and energy consumption. In most cases, the 95% confidence interval

TABLE I
SIMULATION PARAMETERS

Parameter	Values
field size (m)	500 × 500
number of nodes	5000 (dense setting) or 1500 (sparse setting)
communication range (m)	10.0, 20.0,30.0 or 40.0
radius of the monitoring region surrounding a target (m)	50.0
simulation time for each scenario (s)	25.0
maximum speed of a mobile target: v_m (m/s)	2.0 – 20.0
prediction accuracy (p)	0.6 – 1.0
size of a control message ($join$, $prune$, $reconf$) ($byte$)	10
size of the information about a node used in tree expansion and reconfiguration ($byte$)	16 (sequential scheme) or 12 (localized scheme)
size of a sensing report: s_r ($byte$)	40 or 100

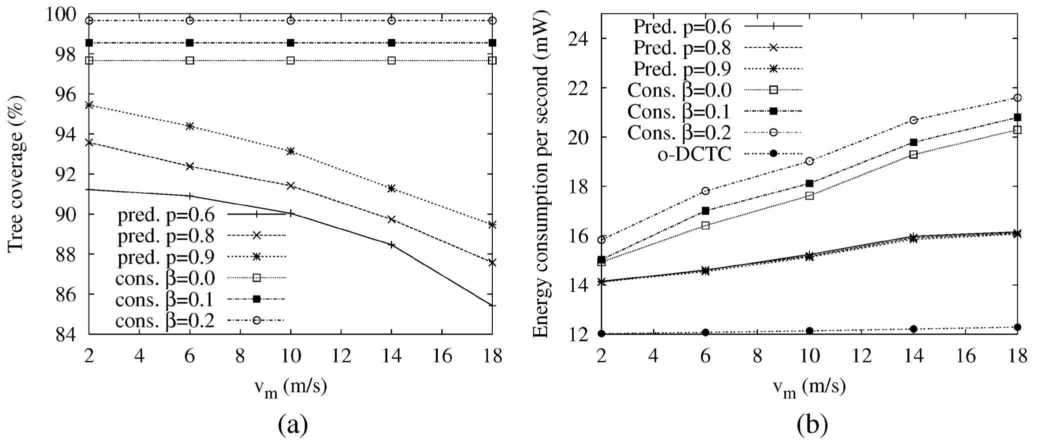

Fig. 8. Comparing the conservative scheme and the prediction-based scheme. (a) Tree coverage. (b) Energy consumption.

for the measured data is less than 6% of the sample mean. Only in some cases (Fig. 10), the 95% confidence interval is about 10% of the sample mean and has been shown in the figure.

1) Comparing the Tree Expansion and Pruning Schemes: We first compare the performance of the conservative scheme, the prediction-based scheme and the o-DCTC scheme in terms of average tree coverage and energy consumption, and then study the impact of some system parameters on the system performance. When comparing the conservative scheme and the prediction-based scheme, the same tree reconfiguration scheme (i.e., the sequential reconfiguration scheme with parameters $d_m = 5$ m and $\alpha = 0.5$) is used.

As shown in Fig. 8(a), the conservative scheme achieves larger tree coverage than the prediction-based scheme. However, as shown in Fig. 8(b), the prediction-based scheme consumes much less energy than the conservative scheme. This is due to the fact that the conservative scheme conservatively expands a convoy tree to include all nodes whose distance to the current target is less than $d_s + v_t + \beta$, whereas the prediction-based scheme relies on prediction to include only a small group of nodes in the tree. With complete knowledge of the network topology and the movement trace of the target, the o-DCTC scheme expands the convoy tree to include and only include the nodes that are within the monitoring region of the target. Thus, it achieves a coverage of 100% and consumes less energy than the other two schemes. Although the o-DCTC scheme has the best performance, it is not practical in real sensor networks. Among the two practical schemes, we prefer the prediction scheme, because it saves lots of energy when compared with the conservative scheme, and it can achieve a high tree coverage and low energy consumption close to the o-DCTC scheme.

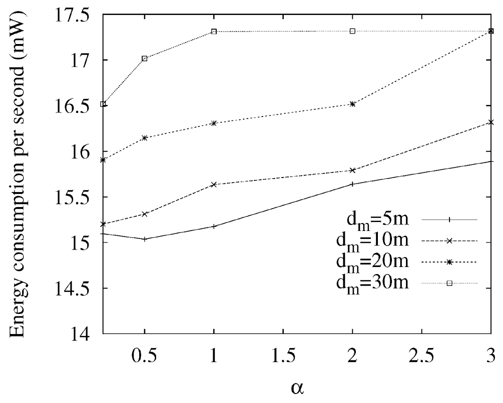

Fig. 9. Fine-tuning the sequential reconfiguration scheme.

Both the tree coverage and the energy consumption of the conservative scheme are affected by parameter β. As described in the conservative scheme, β specifies the scope of velocity

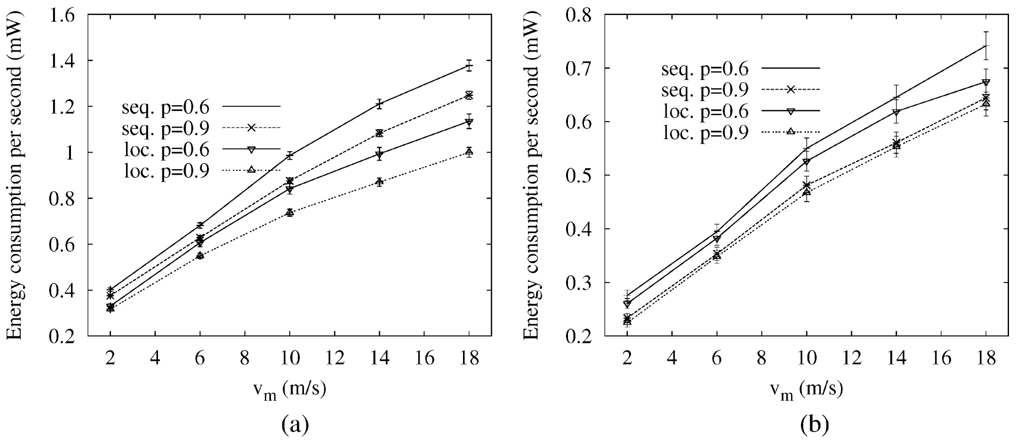

Fig. 10. Energy consumption due to tree expansion and pruning. (a) Dense setting. (b) Sparse setting.

change that can be tolerated by the scheme, and affects the number of nodes that should be included in the tree. If d_s and v_t are fixed, a large β results in a large tree coverage and large energy consumption. As shown in Fig. 8, both the coverage and the energy consumption of the conservative scheme increase as β increases. From the description of the conservative scheme, we know that the velocity of the target is another factor affecting the number of nodes that should be included in the convoy tree. If d_s and β are fixed, the number of nodes in the tree increases as v_t increases. This has been verified in Fig. 8(b), where the energy consumption of the conservative scheme increases as the velocity increases.

Prediction accuracy affects the tree coverage and the energy consumption of the prediction-based scheme. As shown in Fig. 8(b), the tree coverage becomes smaller as the prediction accuracy decreases, since some nodes within the monitoring region of the target may not be included in the tree. Prediction accuracy also affects the energy consumption of the prediction-based scheme. After a misprediction is found, nodes that should not be added to the tree will be pruned, and nodes within the monitoring region but not in the tree will be added. Since these operations increase the energy consumption, as shown in Fig. 8(b), the energy consumption increases as the prediction accuracy drops.

The velocity of the target affects the tree coverage and the energy consumption of the prediction-based scheme. When the movement direction of a target is mispredicted, a larger velocity results in a larger distance between the predicted location and the actual location of the target, and a larger distance in turn results in a larger decrement in the tree coverage. This has been verified by Fig. 8(a), where the tree coverage decreases as the velocity increases. On the other hand, as shown in Fig. 8(b), when the velocity increases, the membership of the tree changes more frequently and the tree needs to be reconfigured more frequently, which increases the energy consumption.

2) Fine-Tuning the Reconfiguration Scheme: Before evaluating the performance of the tree reconfiguration scheme, we need to choose the appropriate values for parameters d_m and α. Since it is difficult to find the values suitable for all scenarios, we tune the parameters to minimize the energy consumption only in some representative scenarios. In these simulations, the prediction-based scheme (with prediction accuracy of 0.8) is used for tree expansion and pruning. The results of fine-tuning the sequential reconfiguration scheme are shown in Fig. 9. We also get similar results when fine-tuning the localized reconfiguration scheme (not shown).

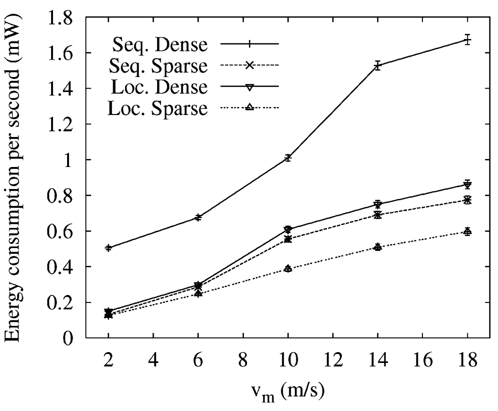

Fig. 11. Comparing the energy consumption due to tree reconfiguration.

Fig. 9 shows the results of tuning the sequential scheme when the maximum velocity of the target is 10 m/s and the sensing report size is 100 B. From Fig. 9, we can see that the energy consumption increases as the parameter d_m increases. This is due to the fact that, when d_m is large, the tree can not be reconfigured (optimized) promptly, resulting in large energy consumption for data collection. Due to the same reason, as shown in the figure, the energy consumption increases as α increases, except when both d_m and α are small. The energy consumption is minimized when d_m and α are both small ($d_m = 5$ m and $\alpha = 0.5$ in this case).

3) Comparing the Reconfiguration Schemes: In this subsection, we compare the performance of the sequential reconfiguration scheme and the localized reconfiguration scheme in terms of energy consumption, using the same tree expansion and pruning scheme (i.e., the prediction-based scheme). As we know, the total energy consumption includes three parts: tree expansion and pruning, tree reconfiguration, and data collection. In the following, we compare the energy consumption of the reconfiguration schemes in detail.

Energy Consumed by Tree Expansion and Pruning: Fig. 10 compares both reconfiguration schemes in

TABLE II
ENERGY CONSUMPTION PER SECOND FOR DATA COLLECTION (mW).

	The Sequential Reconfiguration	The Localized Reconfiguration
Dense, $s_r = 40bytes$	5.137884 ± 0.036342	5.377100 ± 0.038034
Dense, $s_r = 100bytes$	13.137774 ± 0.092927	13.528830 ± 0.095693
Sparse, $s_r = 40bytes$	2.300998 ± 0.048827	2.729444 ± 0.057918
Sparse, $s_r = 100bytes$	5.733158 ± 0.121657	6.577812 ± 0.139580

terms of the energy consumed by tree expansion and pruning. As the velocity increases, more nodes should be added to or pruned from the tree. This increases the energy consumed by tree expansion and pruning. Thus, as shown in Fig. 10, the energy consumption of each scheme increases as the velocity increases. From the figure, we can also see that the energy consumption of the localized scheme is less than the sequential scheme. This is due to the fact that, when adding nodes to the tree, the localized reconfiguration scheme consumes less energy than the sequential reconfiguration scheme, since it requires less information transmitted between grids and less information broadcast within grids. The energy saving becomes more significant when the velocity or the node density increases, since more nodes need to be added to the tree in these two cases.

Energy Consumed by Tree Reconfiguration: Fig. 11 compares both schemes in terms of the energy consumed by tree reconfiguration. When the velocity of the target increases, as shown in Fig. 11, the energy consumed by tree reconfiguration also increases because the tree is reconfigured more frequently. From Fig. 11, we can also see that the localized scheme consumes less energy for tree reconfiguration than the sequential scheme since it requires less information transmitted between grids and less information broadcast within grids.

Energy Consumed by Data Collection: Table II compares both reconfiguration schemes in terms of the energy consumed by data collection. In the sequential reconfiguration scheme, each node is aware of the cost of the nodes in its own grid and its neighboring grids that are close to the root. Based on the knowledge, this scheme guarantees that each node can find an optimal path that has the minimum cost of sending reports to the root. However, when using the localized scheme, a heuristic is used for a node to find its parent node, and this approach can not guarantee that each node can find the optimal path that has the minimum cost of sending reports to the root. Thus, the data collection part consumes less energy in the sequential reconfiguration than that in the localized scheme.

The performance difference between these two schemes is affected by the node density. Under dense setting, the difference is small because the heuristic used in the localized scheme is effective, due to the reasons explained in Section IV.C.III. As shown in Table II, the localized scheme consumes only 3%–5% more energy than the sequential scheme. However, the difference becomes larger under sparse setting, where the localized scheme consumes 15%–20% more energy than the sequential scheme.

Overall Energy Consumption: We now compare both reconfiguration schemes in terms of the overall energy consumption. Under dense setting, as shown in Fig. 12(a) and (b), the performance of the localized scheme is better than the sequential scheme, because the localized scheme significantly outperforms the sequential scheme in terms of the energy consumption for tree expansion and pruning [as shown in Fig. 10(a)] and tree reconfiguration (as shown in Fig. 11). Although the sequential scheme outperforms the localized scheme in terms of the energy consumed by data collection (as shown in Table II), the difference is small. These figures also show that the energy saving becomes larger as the velocity of the target increases, because the energy saving of the localized reconfiguration scheme for tree expansion, pruning and tree reconfiguration increases as the velocity gets larger, and the energy consumed by data collection is not affected by velocity changes.

Fig. 12(c) and (d) show the results under sparse setting. Different from the results under dense setting, the sequential reconfiguration scheme outperforms the localized reconfiguration scheme, because the localized scheme consumes much larger energy for data collection (10%–18%) than the sequential scheme. At the same time, the energy saving of the localized scheme for tree expansion/pruning [as shown in Fig. 10(b)] and tree reconfiguration (as shown in Fig. 11) is small. However, if the velocity is large and the sensing report size is small, as shown in Fig. 12(c), the performance difference between the localized scheme and the sequential scheme becomes less significant, because the localized scheme can save large amount of energy in tree reconfiguration when the velocity is large (as shown in Fig. 11).

VI. Conclusion

In this paper, we defined a new problem called DCTC, which can be applied to detect and track a moving target. We formalized the problem as an optimization problem which needs to find a convoy tree sequence with high tree coverage and low energy consumption. We first proposed an optimal solution based on dynamic programming. Considering the real constraints of a sensor network, we then proposed several practical implementation techniques. For tree expansion and pruning, we proposed two schemes: the conservative scheme and the prediction-based scheme. For tree reconfiguration, the sequential reconfiguration scheme and the localized reconfiguration scheme were presented. Simulation results showed that the prediction-based scheme outperforms the conservative scheme, and it can achieve a relatively high coverage and low energy consumption close to the optimal solution. When the same tree expansion and pruning scheme is used, the localized reconfiguration performs better when the node density is high, and the trend is reversed when the node density is low. The paper defines an important problem and lays out a theoretical foundation. We expect the new problem will spawn a new area of research in the near future.

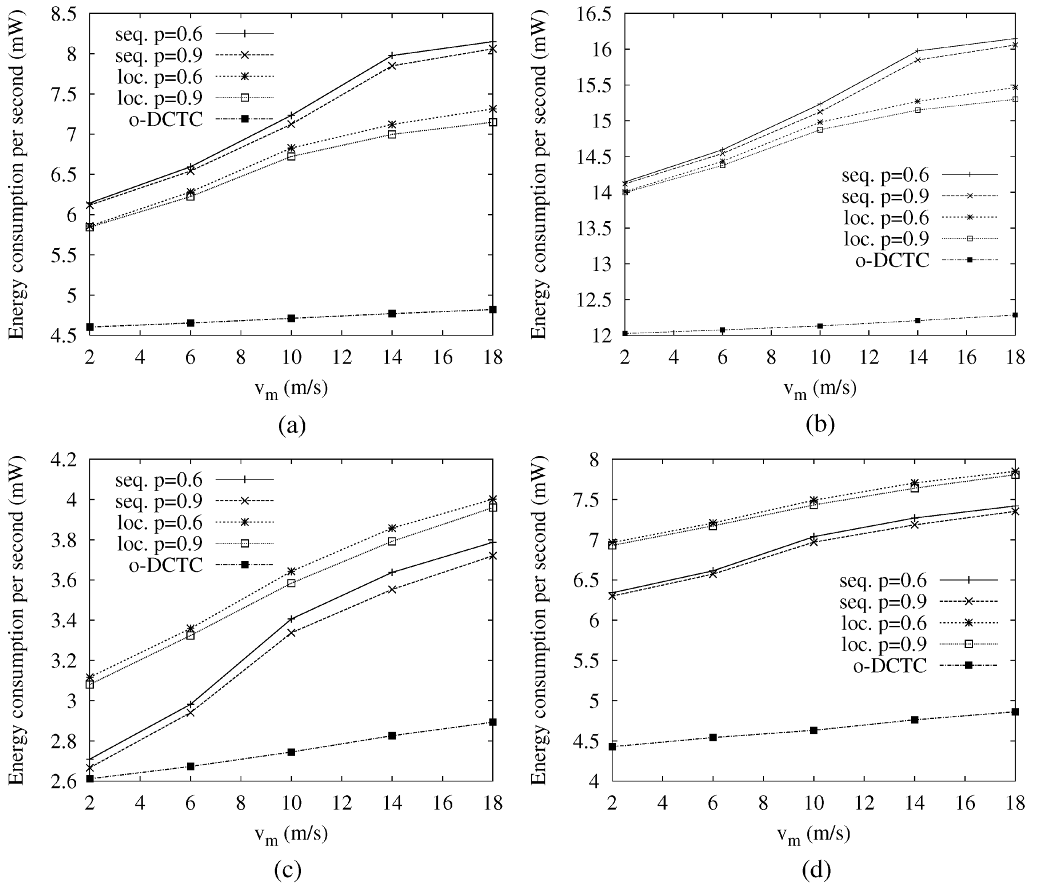

Fig. 12. Comparing the energy consumption of the sequence reconfiguration and the localized reconfiguration.

Acknowledgment

The authors would like to thank the editors and the anonymous referees whose insightful comments helped us to improve the presentation of the paper.

References

[1] "US Naval Observatory (USNO) GPS Operations" (2001, Apr.). [Online]. Available: http://tycho.usno.navy.mil/gps.html

[2] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: A survey," *Comput. Networks*, vol. 38, no. 4, pp. 393–422, Mar. 2002.

[3] A. Aljadhai and T. Znati, "Predictive mobility support for QoS provisioning in mobile wireless environments," *IEEE J. Select. Areas Commun.*, vol. 19, pp. 1915–1930, Oct. 2001.

[4] N. Bulusu, J. Heidemann, and D. Estrin, "GPS-less low cost outdoor location for very small devices," *IEEE Pers. Commun. (Special Issue on Smart Space and Environments)*, vol. 7, pp. 28–34, Oct. 2000.

[5] A. Cerpa, J. Elson, M. Hamilton, and J. Zhao, "Habitat monitoring: Application driver for wireless communications technology," in *Proc. 1st ACM SIGCOMM Workshop Data Communications in Latin America and the Caribbean*, Apr. 2001, pp. 20–41.

[6] M. Chu, H. Haussecker, and F. Zhao, "Scalable information-driven sensor querying and routing for ad hoc heterogeneous sensor networks," *Int. J. High Perform. Comput. Applicat.*, vol. 16, no. 3, pp. 90–110, 2002.

[7] T. Cormen, C. Leiserson, and R. Rivest, *Introduction to Algorithms*. Cambridge, MA: MIT Press, 1990, pp. 514–543.

[8] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocal for wireless microsensor networks," in *Proc. Hawaii Int. Conf. System Sciences*, Jan. 2000, p. 8020.

[9] W. Heinzelman, J. Kulik, and H. Balakrishnan, "Adaptive protocols for information dissemination in wireless sensor networks," in *Proc. Mobile Computing and Networking*, Aug. 1999, pp. 174–185.

[10] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion: A scalable and robust communication," in *Proc. Mobile Computing and Networking*, Aug. 2000, pp. 56–67.

[11] D. Jayasimha, "Fault tolerance in a multisensor environment," in *Proc. 13th Symp. Reliable Distributed Systems*, Oct. 1994, pp. 2–11.

[12] L. Klein, *Sensor and Data Fusion Concepts and Applications*. Bellingham, WA: SPIE Optical Engineering, 1993.

[13] B. Krishnamachari, D. Estrin, and S. Wicker, "Modeling data-centric routing in wireless sensor networks," USC Computer Engineering Tech. Rep. CENG 02–14, 2002.

[14] D. Levine, I. Akyildiz, and M. Naghshineh, "Resource estimation and call admission algorithm for wireless multimedia using the shahow cluster concept," *IEEE/ACM Trans. Networking*, vol. 5, pp. 1–12, Feb. 1997.

[15] G. J. Pottie, *Wireless Sensor Networks*. Killamey, Ireland: ITW, 1998.

[16] The CMU Monarch Project. (1999) The CMU Monarch Project's wireless and mobility extensions to ns. [Online]. Available: http://www.monarch.cs.cmu.edu/cmu-ns.html

[17] I. Stojmenovic and X. Lin, "Power-aware localized routing in wireless networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 12, pp. 1122–1133, Nov. 2001.

[18] G. Tel, *Introduction to Distributed Algorithm*. Cambridge, U.K.: Cambridge Univ. Press, 2000.

[19] S. Tilak, N. Abu-Ghazaleh, and W. Heinzelman, "A taxonomy of wireless micro-sensor network models," *ACM SIGMobile Comput. Commun. Rev. (MC2R)*, vol. 6, pp. 28–36, Apr. 2002.

[20] Y. Xu, J. Heidemann, and D. Estrin, "Geography informed energy conservation for ad hoc routing," in *Proc. ACM Mobile Computing and Networking*, July 2001, pp. 70–84.

[21] Z. Yang and X. Wang, "Joint mobility tracking and hard handoff in cellular networks via sequential Monte Carlo filtering," in *Proc. IEEE INFOCOM*, vol. 2, June 2002, pp. 968–975.

[22] F. Ye, H. Luo, J. Cheng, S. Lu, and L. Zhang, "A two-tier data dissemination model for large-scale wireless sensor networks," in *Proc. ACM Int. Conf. Mobile Computing and Networking*, Sept. 2002, pp. 148–159.

[23] F. Zhao, J. Shin, and J. Reich, "Information-driven dynamic sensor collaboration for tracking applications," *IEEE Signal Processing Mag.*, vol. 19, pp. 68–77, Mar. 2002.

Wensheng Zhang (S'00) received the B.S. degree from Tongji University, Shanghai, China, in 1997 and the M.S. degree in computer science and engineering from the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, in 2000. He is currently working toward the Ph.D degree at The Pennsylvania State University, University Park.

His research interests include wireless ad hoc networks and sensor networks.

Guohong Cao (S'98–A'99) received the B.S. degree from Xian Jiaotong University, Xian, China, and the M.S. degree and the Ph.D. degree in computer science from the Ohio State University, Columbus, in 1997 and 1999, respectively.

He joined the Department of Computer Science and Engineering at The Pennsylvania State University, University Park, in 1999, where he is currently an Associate Professor. His research interests are mobile computing, wireless networks, and distributed fault-tolerant computing. His recent work has focused on data dissemination, cache management, network security, and resource management in wireless networks.

Dr. Cao is an Editor of the IEEE TRANSACTIONS ON MOBILE COMPUTING and IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, has served as a co-chair of the workshop on Mobile Distributed Systems, and has served on the program committee of numerous conferences. He was a recipient of the Presidential Fellowship at the Ohio State University in 1999 and a recipient of the National Science Foundation (NSF) CAREER award in 2001.