Optimizing Tree Reconfiguration for Mobile Target Tracking in Sensor Networks

Wensheng Zhang and Guohong Cao
Department of Computer Science & Engineering
The Pennsylvania State University
University Park, PA 16802
Email: {wezhang, gcao}@cse.psu.edu

Abstract— Sensor nodes have limited sensing range and are not very reliable. To obtain accurate sensing data, many sensor nodes should be deployed and then the collaboration among them becomes an important issue. In [1], a tree-based approach has been proposed to facilitate sensor nodes collaborating in detecting and tracking a mobile target. As the target moves, many nodes in the tree may become faraway from the root of the tree, and hence a large amount of energy may be wasted for them to send their sensing data to the root. In this paper, we address the tree reconfiguration problem. We formalize it as finding a min-cost convoy tree sequence, and solve it by proposing an optimized complete reconfiguration scheme and an optimized interception-based reconfiguration scheme. Analysis and simulation are conducted to compare the proposed schemes with each other and with other reconfiguration schemes. The results show that the proposed schemes are more energy efficient than others.

Index Terms: Reconfiguration, simulations, convoy tree, target tracking, sensor networks.

I. INTRODUCTION

Advances in micro-electro-mechanics and wireless communication have enabled the deployment of large scale sensor networks [2], where thousands of tiny and inexpensive sensor nodes are distributed over a vast field to obtain sensing data. These sensor nodes are equipped with sensing, communicating, and data processing units, which allow sensor nodes to collect, exchange, and process information about the environments. The processing units used in recently designed sensor nodes, e.g., the Medusa MK-2 nodes [3], are already powerful enough to process the sensing data, and will be more powerful in the future. Due to these attractive characteristics, sensor networks become adopted to many military and civil applications such as target tracking, surveillance, environmental control, and security management.

Some limitations of the sensor nodes make sensor network design complicated and intriguing. For example, sensor nodes are not very reliable. They may malfunction or even die out due to energy depletion, and their readings may drift and lose calibration due to environmental interference. Therefore, we cannot rely on a single node to get reliable and accurate results. Because some level of redundancy can be used to deal with failures, multiple nodes can be deployed and collaborate to obtain fine-grain and high-precision sensing data. If the detected target or the monitoring area is large, many sensor nodes may be needed due to limited sensing range. If the detected target is mobile or the monitoring area is dynamic, nodes involved in the collaboration may change over time. The involvement of a large static or dynamic set of senor nodes poses new challenges to design energy efficient sensor networks.

This paper addresses the sensor collaboration issue in target tracking. As shown in Figure 1, the sensor nodes surrounding an adversary tank detect and track the tank, and monitor its surrounding area to count or locate the soldiers in that area. These nodes collaborate among themselves to aggregate data about the tank as well as the surrounding area, and one of them (i.e., the root) generates a fine-grain, high-precision data report. The data report can be saved locally waiting for queries from other nodes [4], or can be forwarded to multiple data centers (the sinks) [5]. Each sink can be a static command center or a moving soldier. As design goals, the sensor nodes surrounding the moving target should promptly detect the target as the target approaches and aggregate their sensing data to generate robust and reliable sensing reports in an energy-efficient way. Also, the network should forward the reports to the sinks in a fast and energy-efficient way.

Most existing researches in sensor networks, e.g., directed diffusion [6], [7], LEACH [8] and TTDD [5], concentrate on finding efficient ways to forward the data report to the data center, and not much work has been done on node collaboration in target tracking. Zhao *et al.* [9], [10], [11] studied the problem of tracking a mobile target using an information-driven approach. In their approach, a single node is used to detect the status of the target. They consider the detection node handoff problem when the target moves, but node collaboration is limited since only a single node is used to track the target at any time. Cerpa *et al.* [12] suggested that multiple nodes surrounding the target should collaborate to make the collected information more complete, reliable and accurate. However, they did not propose any concrete schemes. Brooks *el al.* [13] suggested, but without specifically describing, a cluster-based highly distributed data collection approach, where all nodes (clusters) need to exchange their sensing data to each other.

This method provides fault tolerance, but it is not energy-efficient due to the redundancy in information exchanges.

In our previous work [1], we proposed a *Dynamic Convoy Tree-based Collaboration (DCTC)* framework for target tracking. DCTC relies on a tree structure called *convoy tree*, which includes sensor nodes around the moving target, and the tree dynamically evolves by adding some nodes and pruning some nodes as the target moves. DCTC can significantly reduce the network traffic and the energy consumption by aggregating the sensing data at a node close to the target. A node in the convoy tree only sends data to its parent, which can further reduce the redundancy in data transmission. Certainly, some parent nodes may fail. This can be addressed by allowing a node to select another parent node when its current one fails. If the root fails, a new root will be reselected.

As the target moves, many nodes in the convey tree may become far away from the root, and hence a large amount of energy may be wasted for them to send their sensing data to the root. In this case, a new root should be selected to replace the old root, and the tree should be reconfigured accordingly. In this paper, we address the tree reconfiguration problem. We formalize it as finding a min-cost convoy tree sequence, and solve it by proposing an optimized complete reconfiguration (OCR) scheme and an optimized interception-based reconfiguration (OIR) scheme. Simulation results show that the optimized schemes have better performance than schemes without optimization. The OIR scheme outperforms the OCR scheme when the sensing data size is small or the monitoring region is small, and the trend is reversed in other cases.

The rest of the paper is organized as follows. Section II gives an overview of the DCTC framework and defines the tree reconfiguration problem. In Section III, we present the OCR scheme and the OIR scheme. Section IV evaluates the performance of the proposed schemes, and section V concludes the paper.

II. PRELIMINARIES

A. Assumptions

We consider a sensor network, where sensor nodes are stationary and have a fixed communication range (denoted as d_c). Each node is aware of its own location by GPS [14] or other techniques such as triangulation [15]. Each node also keeps information about its neighbors such as ids and locations. To save power, the sensor nodes stay in sleep most of the time based on the GAF protocol [16]. In this protocol, 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 awake, and other nodes only need to wake up periodically.

Let t_s (t_e) denote the time when the target enters (leaves) the detection region of the network. At any time t ($t_s \leq t \leq t_e$), a set of sensor nodes (denoted as S_t) are required to participate in detecting the target. In this paper, we let S_t include all the nodes whose distance to the target (located at L_t) is less

than a certain *monitoring radius* d_s[1] The circle with a radius of d_s and centered at L_t (see Figure 1) is referred to as the *monitoring region*.

B. Overview of DCTC

In this section, we give an overview of the DCTC framework [1].

1) Constructing the Initial Convoy Tree: When a target first enters the detection region of the sensor network, as shown in Figure 1 (a), nodes that are awake and close to the target can detect it. These nodes construct an initial convoy tree by first selecting a node to be the root of the tree based on a root election algorithm [1]. Then, the other nodes in the monitoring region are added to the convoy tree by selecting the neighbor that has the smallest distance to the root as its parent.

2) Collecting Sensing Data via the Tree: Every certain time interval (1 in this paper), each node in the tree generates a sensing report. Each leaf node sends its data report to its parent. Each intermediate node combines its own data and the sensing data received from its children to form a new report, and sends the report to its parent. Eventually the root receives all the reports and processes them using certain algorithms [13], [17] to generate a final sensing report that will be saved locally, waiting for query or sent to the sinks.

3) Tree Expansion and Pruning: As the target moves, some nodes in the tree become faraway from the target and are pruned. Since most sensor nodes stay sleep to save power before the target arrives, the root should predict the target moving direction and activate the right group of sensor nodes, which can detect the target and monitor its surrounding area as soon as the target approaches. The process of tree expansion and pruning is illustrated in Figure 1 (b). A prediction-based scheme has been proposed in [1] to expand and prune the convoy tree.

As the target moves, many nodes in the tree may become far away from the root, and hence a large amount of energy may be wasted for them to send their sensing data to the root. To reduce the overhead, as shown in Figure 1 (c), 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. In this paper, we address the tree reconfiguration problem.

C. The Problem of Optimizing Tree Reconfiguration

Based on the DCTC framework, we now derive the overall energy consumption for target tracking, and show that the problem of optimizing tree reconfiguration is equivalent to finding a min-cost convoy tree sequence.

We denote the convoy tree at time t ($t_s \leq t \leq t_e$) as $T_t(R)$, where R is the root of the tree. V_t denotes the set of nodes in the tree. Ideally, S_t should be a subset of V_t and all nodes in the monitoring region are in the tree. Let $l(i)$ ($i \in V_t$) be the level of node i, e represent the energy consumed by transmitting a

---

[1]When the node density is very large or the requirement for sensing quality is not very high, S_t can be relaxed to include only a subset of the nodes within the monitoring region.
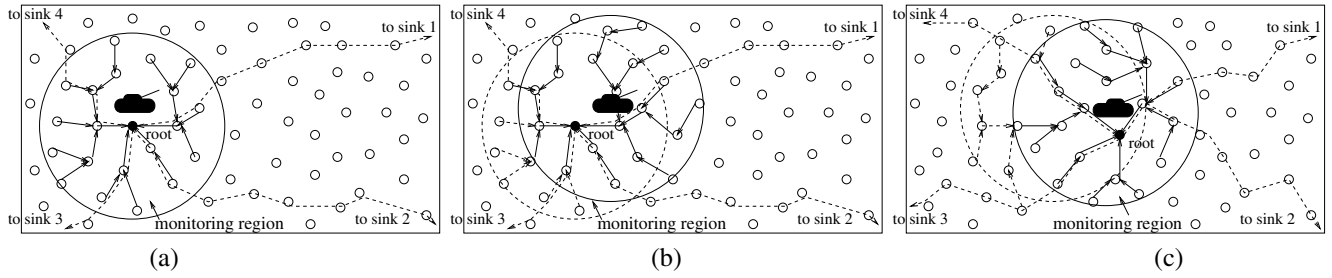
Fig. 1. Using convoy tree to track a target and monitor its surrounding area

unit of data for one hop, and s_d denote the size of sensing data (in units) generated by each node. The energy consumed by the data collection process started at time t is defined as: $E^d(T_t) = e * \sum_{i \in V_t} s_d * l(i)$. By this definition, the energy consumption is proportional to the sum of $l(i)$, because the sensing data generated by each node i should be sent to the root for processing, and the transmission involves $l(i)$ hops.

As trees are reconfigured, a sequence of trees exist at different data collection time between t_s and t_e. These trees form a *convoy tree sequence*, which is denoted as

$$\Gamma(t_s, t_e, T_{t_s}(R_{t_s})) = \langle T_{t_s}(R_{t_s}), T_{t_s+1}(R_{t_s+1}), \cdots, T_{t_e}(R_{t_e}) \rangle$$

Based on the definition of energy consumption for a single convoy tree, we can define the total energy consumption of $\Gamma(t_s, t_e, T_{t_s})$ as follows:

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

where $E^t(T_{t-1}, T_t)$ is the energy consumed by the evolution from T_{t-1} to T_t.

A convoy tree sequence $\Gamma(t_s, t_e, T_{t_s})$ is a *min-cost convoy tree sequence if and only if* $(\forall \Gamma'(t_s, t_e, T_{t_s}))(E(\Gamma(t_s, t_e, T_{t_s})) \leq E(\Gamma'(t_s, t_e, T_{t_s})))$. Since a min-cost convoy tree sequence has the lowest energy consumption among all possible convoy tree sequences, the problem of optimizing tree reconfiguration is equivalent to finding a min-cost convoy tree sequence.

III. OPTIMIZING TREE RECONFIGURATION SCHEMES

A convoy tree is reconfigured in two steps: 1) the current root is replaced by a new one; 2) the remaining part of the tree is reconfigured to reduce the communication overhead. In this section, we first present a basic rule for root replacement, and then propose and analyze two tree reconfiguration schemes.

A. Root Replacement

As illustrated in Figure 2, the root replacement rule is as follows: The current root (R) predicts L_{t+1}, which is the location of the target at the next data collection time, by using certain movement prediction techniques such as [18], [19]. When the distance between R and L_{t+1} is larger than a threshold d_r ($d_r > d_c$), R is replaced by a node closest to L_{t+1}. Once the decision for root replacement is made, R sends a message to the head of the grid that covers L_{t+1}. The
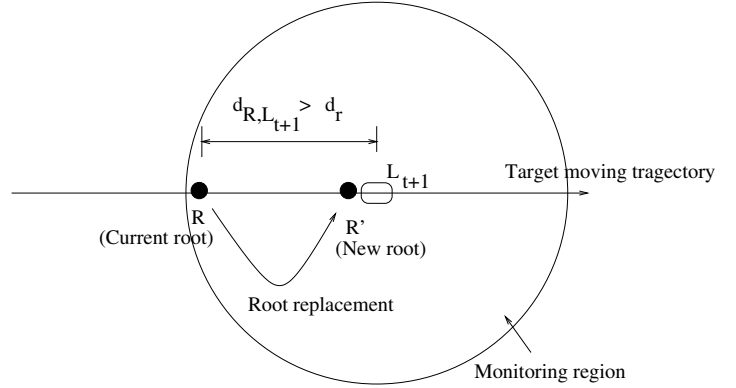

Fig. 2. The root replacement rule

grid head selects the node closest to L_{t+1} among nodes in the grid as the new root. The root replacement rule is based on the following intuitive reasons:

- If the root is close to the target, i.e., the geographic center of the nodes in the tree, the tree should have a short height and small energy consumption during data collection.
- The sensing report may need to be sent to multiple moving sinks [5] distributed in the network, or the data may be stored at the root waiting for queries [4] from multiple sinks located at different locations. In these cases, selecting a root that is faraway from the sensing nodes may not be a good solution, since it may consume lots of network bandwidth and power to send the sensing data to the root.

A Generic Method for Optimizing d_r: The overall energy consumption includes the energy consumed by the data collection part and the tree reconfiguration part. Selecting an appropriate value for d_r is very important. Large d_r may result in high overhead for data collection, and small d_r may result in high overhead for tree reconfiguration. We describe a generic method to compute the optimal d_r to minimize the overall energy consumption. This solution is based on an ideal sensor network model that has following assumptions:

(A1) Nodes are densely and uniformly deployed, and we can always find a node close to a certain location. The node distribution density is denoted as ρ.

(A2) The number of hops between two nodes is proportional to the geographic distance between them. Specif-

ically, the relationship is denoted as $hop(A, B) = \frac{d_{A,B}}{d_c}$, where $d_{A,B}$ is the distance between node A and node B. **(A3)** The target keeps its velocity for a relatively long time before any change.

We consider a time period during which the target moves at velocity v. According to the root replacement rule, root replacement is performed every $k(v) = d_r/v$ time units. Since the sensing data are collected every time unit, the energy consumed for data collection between two successive root replacements is: $\sum_{i=0}^{\lceil k(v) \rceil - 1} \overline{E^d}(i * v)$, where $\overline{E^d}(x)$ represents the data collection overhead when the distance between the root and the target is x. The distance between the old root and the new root is $k(v) * v$ when root replacement is performed, so the energy consumption for tree reconfiguration (initiated by root replacement) is: $\overline{E^t}(k(v) * v)$, where $\overline{E^t}(x)$ represents the tree reconfiguration overhead when the distance between the old root and the new root is x. Therefore, the average energy consumption during this period is:

$$\overline{E}(k(v), v) = \frac{\sum_{i=0}^{\lceil k(v) \rceil - 1} \overline{E^d}(i * v) + \overline{E^t}(k(v) * v)}{k(v)} \quad (1)$$

To minimize $\overline{E}(k(v), v)$,

$$k(v) = arg_{i \in (0, \frac{d_s}{v})} min\{\overline{E}(i, v)\} \quad (2)$$

In the following sections, we present several tree reconfiguration algorithms, and describe how to compute $\overline{E^d}(x)$ and $\overline{E^t}(x)$. With $\overline{E^d}(x)$ and $\overline{E^t}(x)$, $k(v)$ can be computed based on Equation (2). Nodes do not have to compute $k(v)$ on-line. The function can be calculated off-line, and distributed to the related sensor nodes when sensing requests are issued.

B. Optimized Complete Reconfiguration (OCR)

1) Complete Reconfiguration: The basic idea of a complete reconfiguration scheme is as follows: The current root decides and initiates root replacement based on the rules presented in Section III-A. After a root replacement, the new root (R') first broadcasts a message $reconf(R, R')$ to its neighbors. On receiving the message, a node checks if it has received the same message. If so, it ignores the message. Otherwise, it leaves the old tree by detaching from its current parent, and adds to the new tree by attaching to its neighbor that has the smallest distance to the new root. The attach/detach operations help a parent node maintain its children list, and let a child node be synchronized to its parent. The synchronization facilitates data collection, especially for data aggregation at intermediate nodes. If the node is a grid head, it also needs to rebroadcast the message so that nodes out of the communication range of the new root can receive it. This process continues until all nodes within the monitoring region have received the message. Figure 3 illustrates the tree before and after a complete reconfiguration.

2) Overhead Analysis: Based on the ideal sensor network model described in section III-A, we now analyze the energy consumed for data collection and tree reconfiguration in the complete reconfiguration scheme. As shown in Figure 4, the

coordinates of L_t and R are $(0, 0)$ and $(-u, 0)$ respectively. For an arbitrary node P (whose coordinate is (x, y)) within the monitoring region, the energy consumed to send its report to R is:

$$e * s_d * \frac{\sqrt{(x+u)^2 + y^2}}{d_c}.$$

As the node density is ρ, the energy consumed by data collection is:

$$\overline{E^d}(u) = \frac{2 * \rho * e * s_d}{d_c} * \int_{-d_s}^{d_s} \int_0^{\sqrt{d_s^2 - x^2}} \sqrt{(x+u)^2 + y^2} dy dx \quad (3)$$
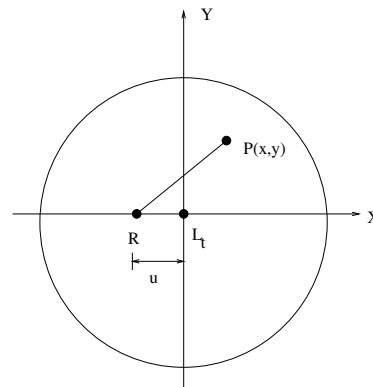

Fig. 4. Analyzing $\overline{E^d}(u)$ for the complete reconfiguration

Next, we analyze $\overline{E^t}(u)$ of the complete reconfiguration scheme. For simplicity, we assume that the reconfiguration occurs exactly when $d_{R, L_{t+1}} = d_r$. In the reconfiguration, each node within the monitoring region sends a message to its old parent to detach from the old tree and sends a message to its new parent to join the new tree. Thus, the energy consumed by tree reconfiguration is estimated by:

$$\overline{E^t}(k(v)) = 2 * \rho * s_c * \pi d_s^2, \quad (4)$$

where s_c is the size of a control message. In this estimation, we ignore the overhead of broadcasting the $reconf$ messages, because the number of broadcasting messages is much less than that of detach/add messages.

3) The OCR Scheme: Figure 5 formally describes the optimal complete reconfiguration (OCR) scheme. In this scheme, a node needs to compute the optimal root replacement threshold ($k(v)$) after it becomes the root. The root uses $k(v) * v$ as the threshold to decide whether to perform root replacement. Sometimes, computing $k(v)$ may be too expensive for a sensor node. In this case, the function can be handed over from the old root. The first root of the tree can obtain the function from the base station from which it receives the sensing task.

Figure 6 gives some examples of $k(v)$ when d_s and the ratio of the data report size to the control message size (s_d/s_c) vary. As can be seen, $k(v)$ is large when v is small, and it decreases as v increases. This can be explained as follows. When the target moves slowly, the set of nodes in the tree changes slowly. In this case, it is not necessary to frequently reconfigure
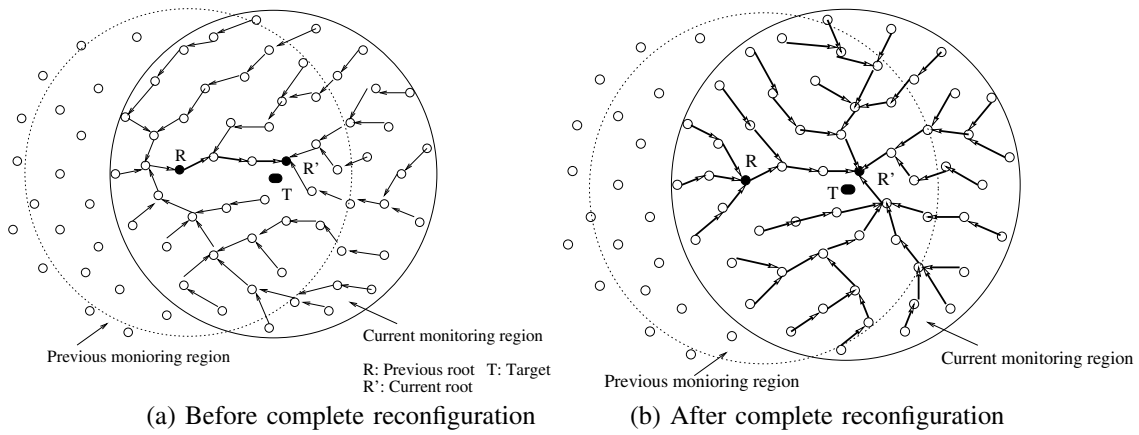
(a) Before complete reconfiguration

R: Previous root T: Target
R': Current root

(b) After complete reconfiguration

Fig. 3.   Illustration of a complete reconfiguration scheme
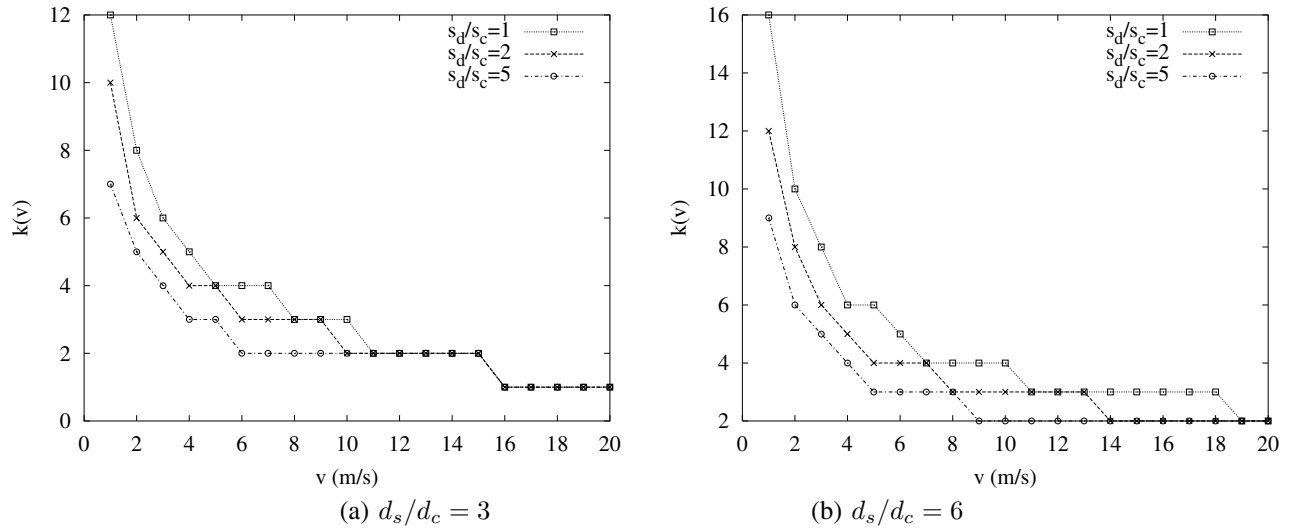

(a) $d_s/d_c = 3$

(b) $d_s/d_c = 6$

Fig. 6.   The optimal $k(v)$ for the complete reconfiguration scheme

the tree due to the reconfiguration overhead. When the target velocity increases, the tree changes quickly. The frequency of tree reconfiguration should be increased to optimize the tree promptly. When v is fixed, $k(v)$ decreases as s_d/s_c increases, because the energy consumption of data collection becomes much higher than tree reconfiguration as s_d/s_c increases. Thus, it is beneficial to increase the tree reconfiguration frequency to reduce the energy consumption of data collection and the overall energy consumption. Comparing Figure 6 (a) with Figure 6 (b), $k(v)$ increases when d_s increases. Because the overhead of tree reconfiguration increases as d_s increases, the threshold for tree reconfiguration is also increased accordingly.

C. Optimized Interception-Based Reconfiguration (OIR)

1) Interception-Based Scheme: Since all nodes in the tree are involved in reconfiguration, the complete reconfiguration scheme may have very high overhead when the tree contains a large number of nodes. To reduce the overhead, we propose an interception-based reconfiguration, which only reconfigures a small part of the tree. Specifically, as shown in Figure 7

(a), only the nodes within the monitoring region and between lines l_0 and l_1 need to change their parents. To simplify the presentation, let the coordinates of the old root (R) and the new root (R') be $(x_0, 0)$ and $(x_1, 0)$ respectively. Lines l_0 and l_1 are defined as follows.

$$l_0 : x = x_1 + d_c \tag{5}$$

$$l_1 : x = x_0 - d_c \tag{6}$$

A node P (whose coordinate is (x, y)) is involved in the reconfiguration if and only if it satisfies:

$$\begin{cases} d_{P,L_t} \leq d_s, \\ x_0 - d_c \leq x \leq x_1 + d_c \end{cases} \tag{7}$$

The interception-based reconfiguration scheme works as follows: After a root replacement, R' first broadcasts a message $reconf(R, R')$ to its neighbors. On receiving the message, the node that satisfies Inequality (7) checks if it has received the message before. If so, the message is ignored. Otherwise, it
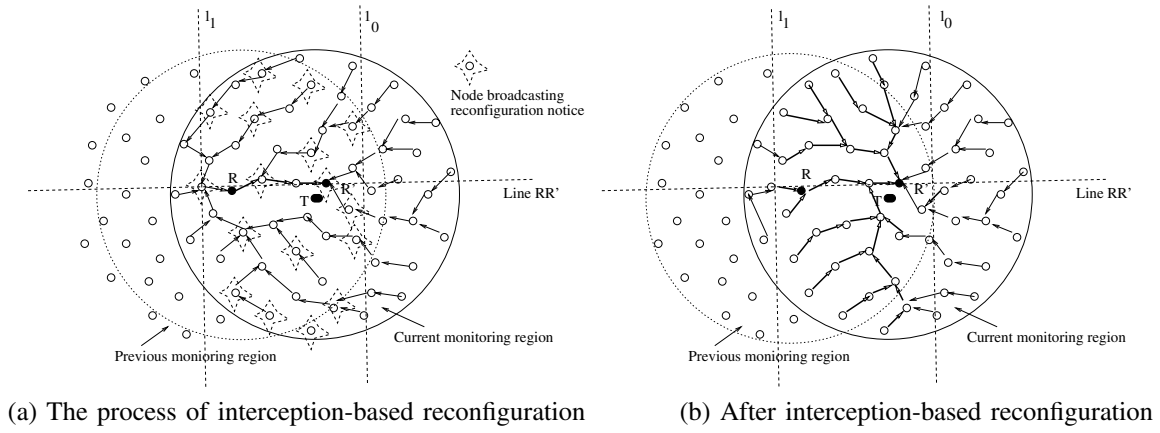
(a) The process of interception-based reconfiguration

(b) After interception-based reconfiguration

Fig. 7. Illustration of the interception-based reconfiguration scheme

leaves the old tree by detaching from its original parent, and adds to the new tree by attaching to its neighbor that has the shortest distance to the new root. If the node is a grid head, it also rebroadcasts the message. The process continues until all nodes satisfying Inequality (7) have received the message. Figure 7 (a) illustrates the reconfiguration process and Figure 7 (b) shows the tree after reconfiguration.
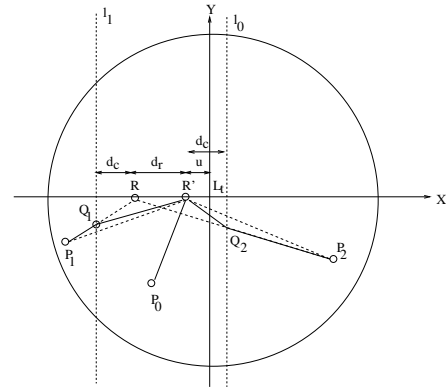

Fig. 8. Analyzing $\overline{E^d}(u)$ of the interception-based reconfiguration scheme

Notations:

- R, R': R is the root of the tree before migration, and R' is the root after migration.
- P_i, C_i: P_i is the parent of node i and C_i is the set of children of i.
- $\mathcal{N}_i$: the set of neighbors of node i.
- $reconf(R, R')$: message to initiate tree reconfiguration.
- $detach(P_i, i)$: message to detach i from P_i.
- $attach(j, i)$: message to attach i to j.

The algorithm executed by root R':

(A) On being selected as the root:
  Compute $k(v)$ from Equations (2), or receive from R;
  Broadcast $reconf(R, R')$;
(B) At data collection time t:
  Monitor the current velocity of the target (v);
  Predict the location of the target at $t + 1$ (L_{t+1});
  if $d_{R', L_{t+1}} > k(v) * v$ **then** do root replacement;

The algorithm executed by node i in the tree:

(A) On receiving $reconf(R, R')$:
  if received $tree\_reconf(R, R')$ before **then** ignore it;
  $j = argmin_{k \in \mathcal{N}_i}\{d_{k, R'}\}$;
  if $P_i \neq j$ **then**
    Send $detach(P_i, i)$ to P_i;
    Send $attach\_req(j, i)$ to j; $P_i = j$;
  if i is a grid head **then**
    Rebroadcast $reconf(R, R')$.
(B) On receiving $attach(i, j)$ from j:
  $C_i = C_i + \{j\}$.
(C) On receiving $detach(i, j)$ from j:
  $C_i = C_i - \{j\}$.

Fig. 5. The OCR Scheme

2) Overhead Analysis: First, we estimate $\overline{E^d}(u)$ of the interception-based reconfiguration. Figure 8 shows the positions of the previous root (R), the current root (R'), and the target (L_t) during a data collection process. Let $d_{R, R'} = d_r$ and $d_{R', L_t} = u$. The coordinates of R, R' and L_t are $(-u - d_r - d_c)$, $(-u, 0)$ and $(0, 0)$ respectively. We divide all nodes in the tree into three parts, and estimate the energy consumed by each part.

For nodes between lines l_0 and l_1:

In Figure 8, an arbitrary node (P_0) locates between l_0 and l_1. With Assumption **(A2)**, $hop(P_0, R) = \frac{d_{P_0, R}}{d_c}$. Thus, the energy consumed to collect data from nodes between l_0 and l_1 is: $\rho * e * s_d * A_0$, where,

$$A_0 = \int_{-u-d_r-d_c}^{-u+d_c} \int_{-\sqrt{d_s^2-x^2}}^{\sqrt{d_s^2-x^2}} \frac{d_{P_0, R}}{d_c} \, dy \, dx \qquad (8)$$

For nodes on the left side of line l_1:
In Figure 8, P_1 is an arbitrary node on the left side of l_1. Since P_1 is out of the region reconfigured by the interception-based scheme, the path between P_1 and R may not be optimized. $hop(P_1, R)$ can be represented as $c_1 * \frac{d_{P_1,R}}{d_c}$, where $c_1 > 1$ is a parameter to be estimated later. The energy consumed to collect data from the nodes on the left side of l_1 is: $\rho*e*s_d*A_1$, where

$$A_1 = \int_{-d_s}^{-u-d_r-d_c} \int_{-\sqrt{d_s^2-x^2}}^{\sqrt{d_s^2-x^2}} c_1 * \frac{d_{P_1,R}}{d_c} \, dy \, dx \qquad (9)$$
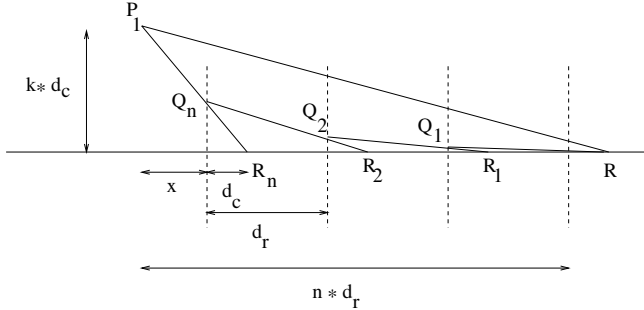
Fig. 9.   The principle of estimating parameter c_1

Figure 9 shows how to estimate c_1. Suppose P_1 was originally on the shortest path to R_n when R_n was the root. After R_n is replaced by R_{n-1}, the path between P_1 and R_{n-1} became $\langle P1, Q_n - R_{n-1}\rangle$. This process continued until R became the root. Thus, the path between P_1 and R is $\langle P1, Q_n, Q_{n-1}, \cdots, Q_2, Q_1, R\rangle$. With assumption **(A2)**

$$hop(P_1, R) = \frac{d_{P_1,Q_n} + \sum_{i=2}^{i=n} d_{Q_i,Q_{i-1}} + d_{Q_1,R}}{d_c}$$

c_1 can be computed by the following equation:

$$c_1 = \frac{hop(P_1, R)}{d_{P_1,R}/d_c}$$

Figure 10 shows the estimated values of c_1. From this figure, we can see that c_1 is small when n is small. With $n*d_r \leq d_s$ and $d_r \geq d_c$, $n \leq d_s/d_c$. Since $d_s \leq 6d_c$ is used in the analysis and simulations, we have $n \leq 6$. Figure 10 shows that c_1 is smaller than 1.1 when $n \leq 6$, so we set c_1 to 1.1 in this paper.
For nodes on the right side of line l_0:
In Figure 8, P_2 is an arbitrary node on the right side of line l_0. Similar to the previous cases, the path between P_2 and R may not be optimized, and $hop(P_2, R)$ can be represented as $c_2 * \frac{d_{P_2,R}}{d_c}$, where $c_2 > 1$ is a parameter. The energy consumed to collect data from nodes on the right side of l_0 is: $\rho*e*s_d*A_2$, where

$$A_2 = \int_{-u+d_c}^{d_s} \int_{-\sqrt{d_s^2-x^2}}^{\sqrt{d_s^2-x^2}} c_2 * \frac{d_{P_2,R}}{d_c} \, dy \, dx \qquad (10)$$

With a method similar to the previous case, we can estimate c_2. We show the estimation results (Due to space limit, we do

not show the estimation detail) in Figure 11, where $n*d_c$ is the distance between R and P_2. Due to the same reason explained in the previous case, we also set c_2 to 1.1.
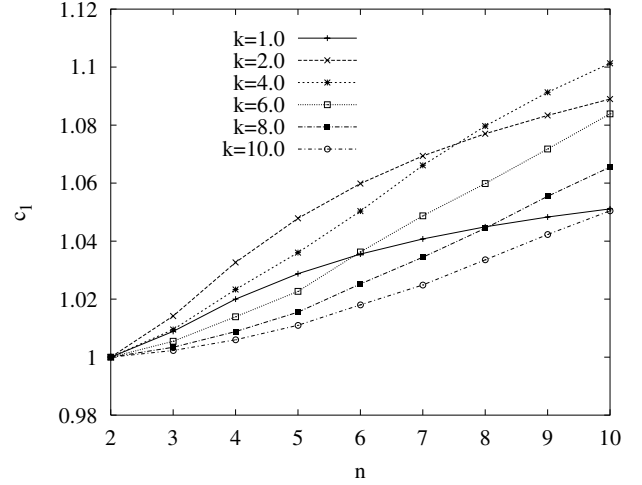

Fig. 11.   Estimated values of c_2

Based on the results of the above three cases, the energy consumed for data collection is:

$$\overline{E^d}(u) = \rho * e * s_d * (A_0 + A_1 + A_2) \qquad (11)$$

where A_0, A_1 and A_2 are defined in Equations (8), (9) and (10).

Next, we estimate the energy consumed for the tree reconfiguration. We assume that the reconfiguration occurs exactly when $d_{R,L_{t+1}} = d_r$. In the reconfiguration, each node within the monitoring region and between line l_0 and line l_1 sends a message to its old parent to detach from the old tree and sends a message to its new parent to add into the new tree. The energy consumed for the reconfiguration process is:

$$\overline{E^t}(d_r) = 2 * \rho * s_c \int_{-u-d_r-d_c}^{d_c} \sqrt{d_s^2 - x^2} \, dx \qquad (12)$$

3) The OIR Scheme: The formal description of OIR is the same as Figure 5 except the following modifications:

- In the algorithm executed by root R', Equations (2) is still used to compute $k(v)$. Although not shown in the formal description, to use Equations (2), $\overline{E^d}(x)$ and $\overline{E^t}(x)$ are calculated by Equations (11) and (12) respectively.
- In Step (A) of the algorithm executed by node i in the tree, add another case to ignore the received $tree\_reconf(R, R')$: **if** it does not satisfy Inequality (7) **then** ignores the message.

Figure 12 shows the optimal value of $k(v)$ when d_s and s_d/s_c change. Due to the same reason explained in Section III-B.3, $k(v)$ decreases as v increases. Compared to Figure 6, the $k(v)$ of OIR is generally smaller than that of OCR. This is because the overhead of the interception-based reconfiguration is much smaller than that of the complete reconfiguration and
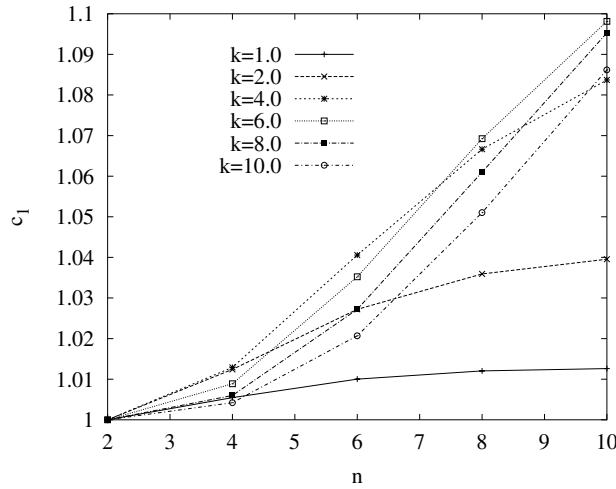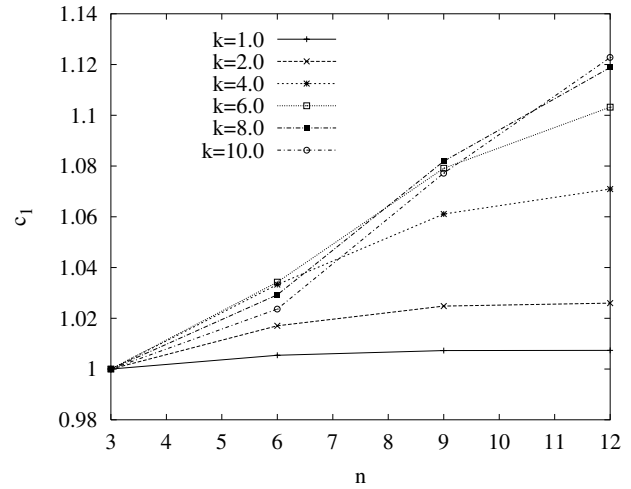
(a) $d_r = d_c$ (b) $d_r = 2d_c$

Fig. 10. Estimated values of c_1
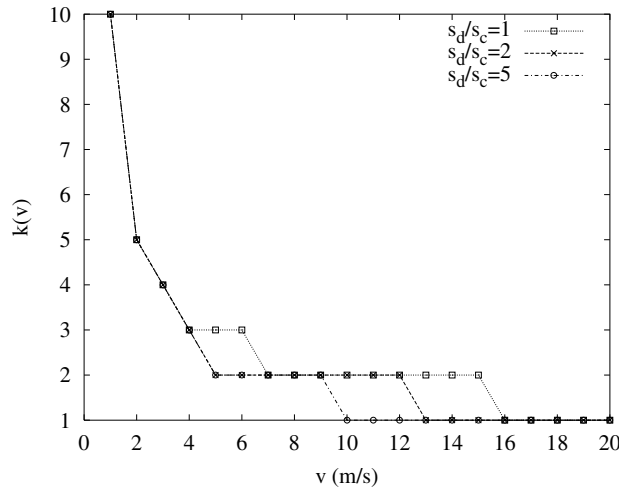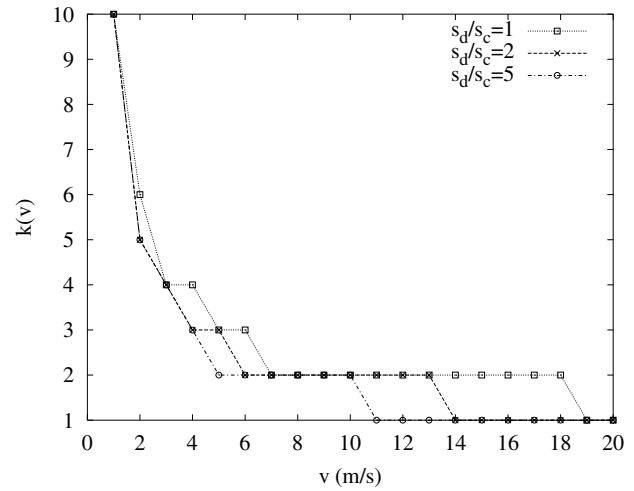
(a) $d_s/d_c = 3$ (b) $d_s/d_c = 6$

Fig. 12. The optimal $k(v)$ for the interception-based reconfiguration scheme

it is beneficial for OIR to reconfigure the tree more frequently than OCR.

D. Analytical Comparisons between OCR and OIR

OCR and OIR take different approaches to optimize the overall energy consumption: OCR gives higher priority to data collection while OIR gives higher priority to tree reconfiguration. It is important to compare these schemes to find the best scenarios for each of them. We compute the overall energy consumption of OCR from Equations (1), (3) and (4), and that of OIR from Equations (1), (11) and (12). The results are shown in Figure 13. Note that the "Energy Consumption" shown in the figure is the ratio of the estimated amount of energy consumption to a common reference value.

Figure 13 shows that the energy consumption increases as the velocity of the target increases. This is because the reconfiguration frequency increases as the velocity increases

(shown in Figure 6 and 12), and then the energy consumption for reconfiguration also increases. Since OIR has smaller reconfiguration overhead than OCR, the energy consumption of OIR is shown to increase more slowly. As a result, OIR outperforms OCR when the velocity is high.

Figure 13 also shows the impact of s_d/s_c. Compared to OCR, OIR reduces the reconfiguration overhead at the cost of increasing the data collection overhead. If the data size is small (e.g., $s_d/s_c = 1$), the increase of the data collection overhead in OIR is also small. Therefore, OIR outperforms OCR. As the data size increases (e.g., $s_d/s_c = 6$), the data collection overhead in OIR also increases, and it may exceed the saving in reconfiguration overhead when the reconfiguration frequency is low (i.e., the velocity is small).

Comparing Figure 13 (a) and (b), we can see the impact of d_s/d_c on the performance. Since OIR reconfigures only a

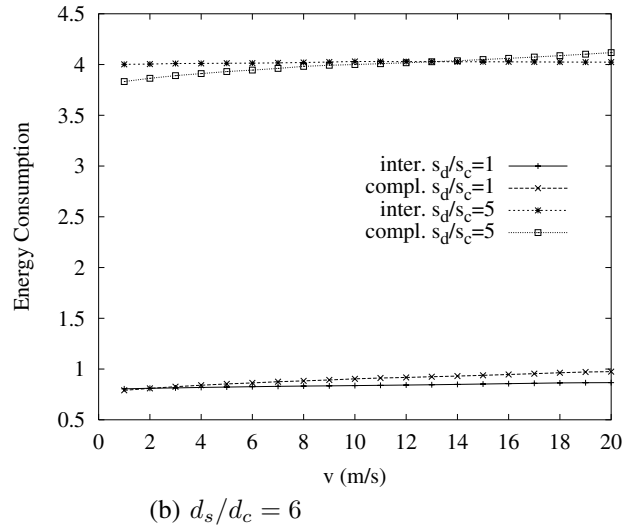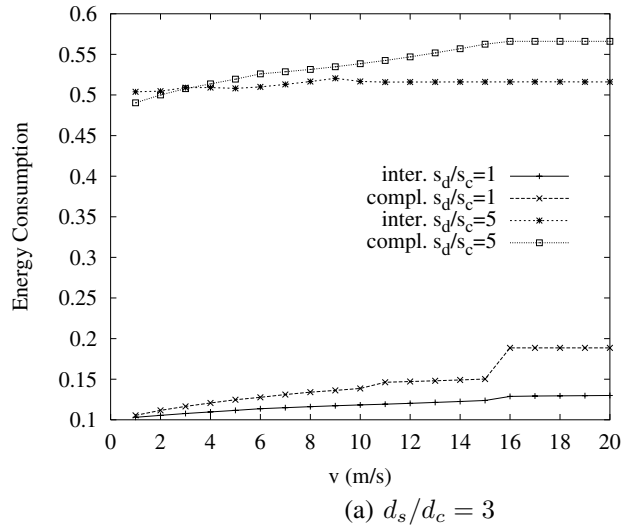(a) $d_s/d_c = 3$ (b) $d_s/d_c = 6$

Fig. 13. Analytical comparisons between OCR and OIR

small part of the tree, the increased overhead in data collection increases as the size of the monitor region (i.e., d_s/d_c) increases. As a result, the overall benefit of OIR becomes smaller as d_s/d_c increases.

In summary, OIR outperforms OCR when the velocity is large, s_d/s_c is small, or d_s/d_c is small. The trend is reversed in other cases.

IV. PERFORMANCE EVALUATIONS

A. Simulation Model

We developed a simulator based on $ns2$ (version 2.1b8a) [20], to evaluate the performance of the proposed schemes, and compare them to other non-optimization reconfiguration schemes listed in Table I. In this simulator, the MAC protocol is based on IEEE 802.11, and the two-ray ground propagation model is adopted. The transmission range of each node is fixed at $20m$. Sensor nodes are distributed over a $400 \times 400 m^2$ flat field, which is divided into $9 \times 9 m^2$ grids. In each experiment, 6000 nodes are deployed and the deployment guarantees that there are at least three nodes in each grid.

TABLE I

NON-OPTIMIZED RECONFIGURATION SCHEMES

Name	Characteristics
Aggressive Complete Reconfiguration (ACR)	A complete reconfiguration is initiated when $d_{R,L_{t+1}} \geq d_c$
Conservative Complete Reconfiguration (CCR)	A complete reconfiguration is initiated when $d_{R,L_{t+1}} \geq d_s$
Aggressive Interception-based Reconfiguration (AIR)	An interception-based reconfiguration is initiated when $d_{R,L_{t+1}} \geq d_c$
Conservative Interception-based Reconfiguration (CIR)	An interception-based reconfiguration is initiated when $d_{R,L_{t+1}} \geq d_s$

We use a mobility model similar to [18] to simulate the movement of the target. At the beginning of the simulation, the target shows up at a random location on the border of the field with an initial moving direction and velocity. Its

moving direction can be one of the following: north (N), northeast (NE), east (E), southeast (SE), south (S), southwest (SW), west (W) and northwest (NW). Its velocity is uniformly distributed between 0 and v_m. Every $10s$, the target may change its moving direction and/or velocity. Specifically, with a probability of p_k, the direction and velocity of the target keep unchanged. With a probability of $p'_k(p'_k = 0.25(1 - p_k))$, the moving direction will change 45° or 90°, and the velocity will be reselected from $[0, v_m]$. The monitoring range surrounding a target is a circle with the radius of $30.0m$ or $60.0m$. Table II lists most of the simulation parameters.

TABLE II

SIMULATION PARAMETERS

Parameter	Values
field size (m^2)	$400.0 * 400.0$
number of nodes	6000
communication range (m): d_c	20.0
monitoring radius (m): d_s	30.0, 60.0
size of data report $(byte)$: s_d	10,50
size of control message $(byte)$: d_c	10
maximum velocity of a mobile target (m/s): v_m	$[1.0, 20.0]$
probability that the mobile target keeps the same velocity: p_k	$[0.6, 0.9]$
data collection interval (s)	1.0

B. Simulation Results

1) Energy Consumption: Figure 14 shows the energy consumption as a function of the maximum velocity of the moving target. When the velocity increases, the tree reconfiguration frequency increases and the energy consumption also increases.

We first study the impact of root replacement threshold on energy consumption. Figure 14 (a) shows that OCR has the best performance among the complete reconfiguration schemes, followed by ACR and CCR. In the interception-based schemes, OIR outperforms AIR which outperforms
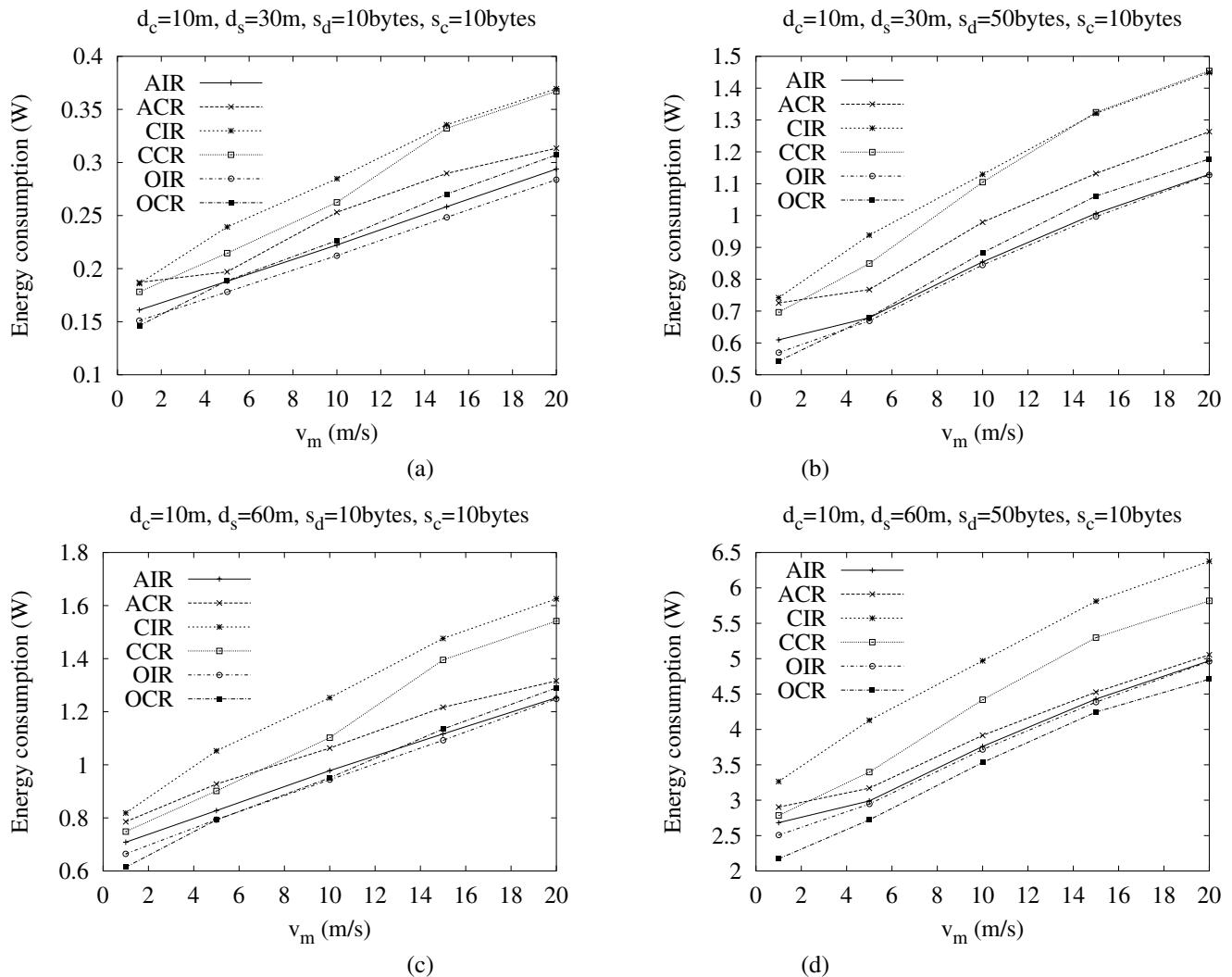
Fig. 14. Comparing the energy consumption of different reconfiguration schemes ($p_k = 0.8$)

CIR. This is due to the following reasons. As the target moves, promptly reconfiguring the tree can reduce the energy consumption for data collection. Thus, ACR outperforms CCR and AIR outperforms CIR. However, frequent reconfigurations may introduce high overhead. OCR and OIR reduce unnecessary reconfigurations by using the optimal root replacement threshold, and hence outperform ACR and AIR respectively. Figure 14 (b), (c) and (d) show similar results, except that the difference between AIR (ACR) and CIR (CCR) is larger when s_d or d_s increases. This is because the energy consumption of data collection becomes much higher than that of tree reconfiguration, and then the aggressive schemes outperform the conservative schemes.

We now compare the performance of the reconfiguration schemes that use the same root replacement threshold. Figure 14 (a) shows that CCR outperforms CIR due to the following reasons. Since the tree reconfiguration is not very frequent in these schemes, the energy consumption of data collection dominates the overall energy consumption. CIR optimizes only part

of the tree, and hence using CIR has larger energy consumption for data collection compared to using CCR. When the velocity increases, the frequency for tree reconfiguration also increases. This will reduce the performance difference between CIR and CCR, since CIR has smaller energy consumption for tree reconfiguration. Comparing 14 (a) with (b), (c) and (d), we can find that the difference between CIR and CCR increases as s_d (or d_s) increases, because the energy consumed for data collection increases.

Figure 14 (a) also shows that AIR outperforms ACR. In both schemes, the trees are reconfigured very frequently, so there is no significant difference in the energy consumed by data collection. However, AIR consumes less energy for tree reconfiguration than ACR. The same trend is shown in Figure 14 (b), (c), (d), and the difference between AIR and ACR increases as s_d (or d_s) increases, since the energy consumption of tree reconfiguration also increases.

By comparing OIR and OCR in different scenarios, we have the following observations: OIR outperforms OCR when s_d/s_c
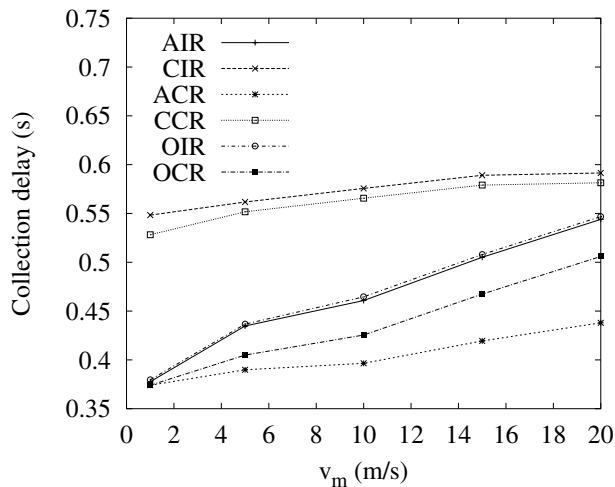
Fig. 15. Comparing the collection delay of different reconfiguration schemes ($d_s = 30m$, $s_d = 10bytes$)

and d_s/d_c are small. The difference between OIR and OCR becomes smaller as s_d/s_c or d_s/d_c increases. With a large s_d/s_c and d_s/d_c, OCR outperforms OIR. These results are consistent with the analytical results.

2) Data Collection Delay: Figure 15 shows the average data collection delay as a function of velocity. Since tree reconfiguration is only done at certain interval, the tree may not be reconfigured promptly if the moving velocity is high. Thus, the data collection delay is shown to be slightly higher when the velocity increases. The root replacement method has significant effects on the data collection delay. Since frequent reconfiguration can promptly reduce the height of a tree, ACR (AIR) has smaller delay than CCR (CIR). When the same root replacement scheme is used, the data collection delay is also affected by the tree reconfiguration method. Compared to complete reconfiguration, interception-based reconfiguration changes only a small part of the tree, and the resulted tree has a larger height. Thus, AIR (CIR) has longer delay than ACR (CCR). Due to the same reason, OIR also has longer delay than OCR. Among all these schemes, the data collection delay of OIR and OCR are not optimal, but the difference to the optimal value is reasonable.

3) Impact of Movement Predication Accuracy: Figure 16 shows the impact of p_k on the energy consumption of OIR and OCR. As shown in the figure, the energy consumption increases as p_k drops. As p_k drops, the chance of wrong prediction increases, and then the computed root replacement threshold may not be optimal. A wrong prediction may cause the root to migrate to a wrong direction, and another root replacement has to be performed, increasing the reconfiguration overhead. However, the energy consumption does not increase too much if p_k is not very low.

V. CONCLUSIONS

This paper studied the problem of optimizing tree reconfiguration when the target moves. We formalized it as finding a min-cost convoy tree sequence, and solved it by proposing an optimized complete reconfiguration (OCR) scheme and an optimized interception-based reconfiguration (OIR) scheme. These two schemes are based on two classes of reconfiguration schemes: the complete reconfiguration which aims to minimize the energy consumption for data collection, and the interception-based reconfiguration which aims to reduce the reconfiguration overhead. OCR and OIR optimize these two classes of schemes by selecting appropriate root replacement threshold to minimize the overall energy consumption. Extensive analysis and simulations are conducted to compare the performance of the optimized schemes and some other schemes. The results show that the optimized schemes have better performance than other schemes. OIR outperforms OCR when the size of the sensing data is small or the monitoring region is small, and the trend is reversed in other cases.

REFERENCES

[1] W. Zhang and G. Cao, "DCTC: Dynamic Convoy Tree-Based Collaboration for Target Tracking in Sensor Networks," *IEEE Transactions on Wireless Communication, in press, also available at: http://www.cse.psu.edu/~gcao*.
[2] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless Sensor Networks: A Survey," *Computer Networks*, vol. 38, no. 4, March 2002.
[3] A. Savvides and M. Srivastava, "A Distributed Computation Platform for Wireless Embedded Sensing," *International Conference on Computer Design (ICCD)*, September 2002.
[4] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan an d S. Shenker, "GHT: A Geographic Hash Table for Data-Centric Storage," *WSNA '02*, September 2002.
[5] F. Ye, H. Luo, J. Cheng, S. Lu, and L. Zhang, "A Two-Tier Data Dissemination Model for Large-scale Wireless Sensor Networks," *ACM MOBICOM'02*, pp. 148–159, September 2002.
[6] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed Diffusion: A Scalable and Robust Communication," *ACM MobiCOM '00*, August 2000.
[7] B. Krishnamachari, D. Estrin, and S. Wicker, "Modelling Data-Centric Routing in Wireless Sensor Networks," *IEEE INFOCOM'02*, June 2002.
[8] W. Heinzelman, J. Kulik, and H. Balakrishnan, "Adaptive Protocols for Information Dissemination in Wireless Sensor Network," *ACM Mobicom'99*, August 1999.
[9] M. Chu, H. Haussecker and F. Zhao, "Scalable Information-driven Sensor Querying and Routing for Ad Hoc Heterogeneous Sensor Networks," *International Journal of High Performance Computing Applications*, 2002.
[10] J. Liu, J. Liu, L. Reich, P. Cheung, and F. Zhao, "Distributed Group Management for Track Initiation and Maintenance in Target Localization Applications," *2nd Workshop on Information Processing in Sensor Networks (IPSN)*, April 2003.
[11] F. Zhao,J. Shin and J. Reich, "Information-driven Dynamic Sensor Collaboration for Tracking Applications," *IEEE Signal Processing Magazine*, pp. 68–77, March 2002.
[12] A. Cerpa, J. Elson, M. Hamilton, and J. Zhao, " Habitat Monitoring: Application Driver for Wireless Communications Technology," *First ACM SIGCOMM Workshop on Data Communications in Latin America and the Caribbean*, April 2001.
[13] B. Brooks,C. Griffin and D. Friedlander, "Self-Organized Distributed Sensor Network Entity Tracking," *International Journal of High Performance Computing Applications, Special Issue on Sensor Networks*, 2002.
[14] "US Naval Observatory (USNO) GPS Operations," *http://tycho.usno.navy.mil/gps.html*, April 2001.
[15] N. Bulusu, J. Heidemann, and D. Estrin, "GPS-less Low Cost Outdoor Location For Very Small Devices," *IEEE Personal Communication, Special Issue on "Smart Space and Environments"*, October 2000.
[16] Y. Xu, J. Heidemann and D. Estrin, "Geography Informed Energy Conservation for Ad Hoc Routing," *ACM MOBICOM'01*, July 2001.
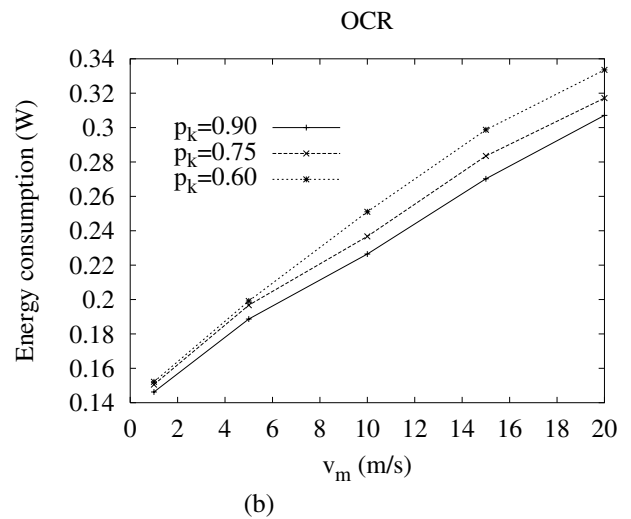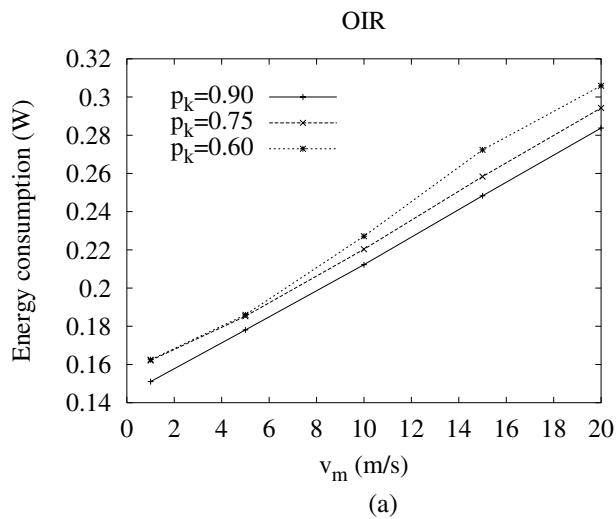
Fig. 16. The impact of p_k on energy consumption ($d_s = 30m$, $s_d = 10bytes$)

[17] D. Li, K. Wong, Y. Hu and A. Sayeed, "Detection, Classification and Tracking of Targets in Distributed Sensor Networks," *IEEE Signal Processing Magazine*, pp. 17–29, March 2002.

[18] A. Aljadhai and T. Znati, "Predictive Mobility Support for QoS Provisioning in Mobile Wireless Environments," *IEEE Journal on Selected Areas in Communications*, vol. 19, no. 10, October 2001.

[19] Z. Yang and X. Wang, "Joint Mobility Tracking and Hard Handoff in Cellular Networks via Sequential Monte Carlo Filtering," *IEEE INFOCOM'02*, June 2002.

[20] The CMU Monarch Project, "The CMU Monarch Projects Wireless and Mobility Extensions to ns," *http://www.monarch.cs.cmu.edu/cmu-ns.html*, October 1999.