Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2013, Article ID 494863, 16 pages
http://dx.doi.org/10.1155/2013/494863

Research Article

A Hybrid Cluster-Based Target Tracking Protocol for Wireless Sensor Networks

Zhibo Wang,[1] Wei Lou,[2,3] Zhi Wang,[1] Junchao Ma,[3] and Honglong Chen[4]

[1] *State Key Laboratory of Industrial Control Technology, Zhejiang University, Hangzhou 310027, China*
[2] *Shenzhen Research Institute, The Hong Kong Polytechnic University, Shenzhen 518057, China*
[3] *Department of Computing, the Hong Kong Polytechnic University, Hong Kong*
[4] *College of Information and Control Engineering, China University of Petroleum, Qingdao 266580, China*

Correspondence should be addressed to Zhi Wang; wangzhi@iipc.zju.edu.cn

Received 11 November 2012; Revised 15 February 2013; Accepted 7 March 2013

Academic Editor: Tian He

Target tracking is a typical and important application of wireless sensor networks (WSNs). In consideration of the network scalability and energy efficiency for target tracking in large-scale WSNs, it has been employed as an effective solution by organizing the WSNs into clusters. However, tracking a moving target in cluster-based WSNs suffers a boundary problem when the target moves across or along the boundaries of clusters, as the static cluster membership prevents sensors in different clusters from sharing information. In this paper, we propose a novel mobility management protocol, called *hybrid cluster-based target tracking* (HCTT), which integrates on-demand dynamic clustering into a cluster-based WSN for target tracking. By constructing on-demand dynamic clusters at boundary regions, nodes from different static clusters that detect the target can temporarily share information, and the tracking task can be handed over smoothly from one static cluster to another. As the target moves, static clusters and on-demand dynamic clusters alternately manage the target tracking task. Simulation results show that the proposed protocol performs better in tracking the moving target when compared with other typical target tracking protocols.

1. Introduction

Target tracking is considered important in WSNs, as it is a base for many practical applications, such as battlefield surveillance, emergency rescue, disaster response, and patient monitoring [1]. Generally speaking, target tracking aims to detect the presence of a target and compute reliable estimates of its locations, while the target moves within an area of interest and forward these estimates to the base station in a timely manner.

It is known that the cluster structure can provide benefits for large-scale WSNs. For example, it facilitates spatial reuse of resources to increase the system capacity [2]; it also benefits local collaboration and routing [3, 4]. Recently, the cluster structure is gradually adopted for solving the target tracking problem [5–13]. Normally, nodes surrounding the target collaborate with each other to estimate the location of the target. In [5–10], dynamic clustering approaches are used that dynamically wake up a group of nodes to construct a cluster for local collaboration when the target moves into a region. Clusters are constructed dynamically, as the target moves. Dynamic clustering is obviously an efficient way for local sensor collaborations because clusters formed at each time instant change dynamically, as the target moves. However, the dynamic clustering process, repeating as the target moves, is energy costly, as it incurs much overhead for forming and dismissing clusters. Besides, dynamic clustering does not consider how to efficiently send data to the sink, which is another important aspect of target tracking.

In contrast, the target tracking in [11–13] uses the static cluster for the network scalability and energy efficiency (the term of "static cluster" does not mean that the cluster will not change during the network's entire lifetime. It means that the cluster structure will keep unchanged for a relatively longer period of time compared to a temporally formed dynamic cluster, until the next round of clustering process begins. In this way, as shown in the LEACH protocol [14], each sensor node has the probability of becoming a cluster

head so as to balance the energy load). It uses a predictive mechanism to inform cluster heads about the approaching target, and then the corresponding cluster head wakes up a number of appropriate nodes right before the arrival of the target. The overhead can be saved as the tracking task is handed over from one static cluster to another without costly dynamic clustering processes. It also provides a scalable structure for coordinating and managing networks. Therefore, static cluster-based approaches are more suitable for target tracking in large-scale sensor networks. However, the static cluster membership prevents sensors in different clusters from collaborating and sharing information with each other, which causes a so-called *boundary problem* when the target moves across or along the boundaries of clusters. The boundary problem will result in the increase of tracking uncertainty or even the loss of the target. Therefore, a new protocol is required to solve the *boundary problem* and realize the tradeoff between energy consumption and local sensor collaboration for cluster-based sensor networks.

In this paper, we propose a novel distributed mobility management protocol, called *hybrid cluster-based target tracking* (HCTT), for efficient target tracking in a large-scale cluster-based WSN. HCTT integrates on-demand dynamic clustering into a scalable cluster-based WSN with the help of boundary nodes, which facilitates sensors' collaboration among clusters to solve the boundary problem. As shown in Figure 1, when the target is inside a static cluster, the cluster is responsible for target tracking; as the target moves close to the boundaries of clusters, an on-demand dynamic clustering process will be triggered to manage the tracking task so as to avoid the boundary problem. The on-demand dynamic cluster will dismiss soon after the target moves away from the boundaries. As shown in Figure 1, the sequential clusters for tracking the target are $A \rightarrow D1 \rightarrow C \rightarrow D2 \rightarrow E$. When the target moves, static clusters and on-demand dynamic clusters alternately manage the tracking task. By integrating on-demand dynamic clustering into a scalable cluster-based structure, nodes belonging to different static clusters can share information, which guarantees smoothly target tracking and realizes well tradeoff between energy consumption and local sensor collaboration. Note that a conference paper [15] containing some preliminary results of this paper appeared at IEEE DCOSS 2010. This paper has been revised by following the comments of the reviewers and the feedback of other researchers. Moreover, we analyze the computation and communication complexity of HCTT and extend the performance evaluation to multihop cluster scenario as opposed to just one-hop scenario.

The main contributions of this paper are summarized as follows.

(i) We address the boundary problem for target tracking in a cluster-based WSN.

(ii) We present a novel hybrid cluster-based target tracking protocol, which balances well between the energy consumption and local sensor collaboration.

(iii) The proposed algorithm is scalable to multi-hop static clusters and solves the problem of tracking uncertainty due to prediction errors.

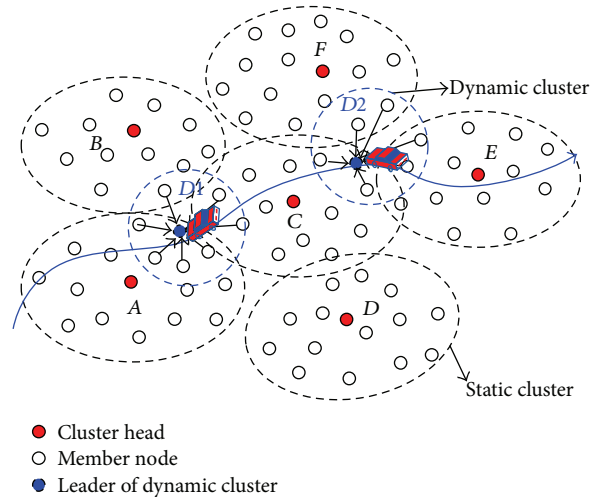

Figure 1: Illustration of HCTT for target tracking in a cluster-based WSN.

(iv) We conduct simulations to show the efficiency of the proposed scheme compared with other typical target tracking solutions.

The rest of the paper is organized as follows. Section 2 reviews the related work about target tracking in WSNs. Section 3 describes the system model and the boundary problem for target tracking in a cluster-based WSN. In Section 4, we present the hybrid cluster-based target tracking protocol in detail. The analysis of HCTT is presented in Section 5. Section 6 demonstrates the performance evaluation for one-hop clusters and multi-hop clusters. Finally, we conclude the paper in Section 7.

2. Related Work

Target tracking in WSNs has received considerable attention from various angles, and a lot of protocols have been proposed. Zhao et al. proposed an information-driven dynamic sensor collaboration mechanism for target tracking [16]. Brooks et al. presented a distributed entity-tracking framework for sensor networks [17]. Chen et al. proposed distributed sensor scheduling algorithms for binary sensor networks [18] and acoustic sensor networks [19], respectively. Wang et al. proposed a novel localization algorithm for target tracking [20]. Vigilnet, an energy-efficient and real-time integrated system for target tracking, was designed and implemented in [21, 22]. A distributed TDMA scheduling algorithm for target tracking in ultrasonic sensor networks was proposed in [23]. In [24], the authors proposed a secure localization scheme to defend against some typical attacks that may affect target tracking accuracy. More target tracking protocols can be found in [25, 26].

To balance energy consumption and sensor collaboration, a lot of dynamic clustering protocols have been proposed for target tracking in WSNs. Yang et al. proposed an adaptive dynamic cluster-based tracking (ADCT) protocol

that dynamically selects cluster heads and wakes up nodes to construct clusters with the help of a prediction algorithm, as the target moves in the network [6]. Zhang and Cao proposed a dynamic convoy tree-based collaboration (DCTC) framework to detect and track the mobile target [7]. The convoy tree can be considered as a cluster which is dynamically configured by adding and pruning some nodes, as the target moves. Ji et al. proposed a dynamic cluster-based structure for object detection and tracking [8]. Jin et al. also proposed a dynamic clustering mechanism for target tracking in WSNs that balances the missing rate and energy consumption [9]. Medeiros et al. proposed a dynamic clustering algorithm for target tracking in wireless camera Networks [10]. A decentralized dynamic clustering protocol for acoustic target tracking, which relies on a static backbone of sparsely placed high-capacity sensors, was proposed in [5]. As mentioned in Section 1, dynamic clustering is efficient for local sensor collaboration but incurs too much overhead for cluster construction and dismiss, as the target moves. In this paper, HCTT overcomes the clustering overhead by relying on a preexisting static cluster structure.

Several target tracking protocols are based on static cluster structure [11–13]. Sensor nodes are organized into clusters by using suitable clustering protocols, such as LEACH [14] and HEED [27]. With the cluster structure, Yang and Sikdar proposed a distributed predictive tracking (DPT) protocol that predicts the next location of the target and informs the cluster head about the approaching target. The corresponding cluster head then wakes up the closet three sensors nodes around the predicted location before the arrival of the target [11]. Wang et al. [12] proposed a hierarchical prediction strategy (HPS) that also relies on cluster structure for target tracking and implemented a real target tracking system in [13]. Compared with dynamic clustering protocols, cluster-based target tracking protocols take the advantages of underlying cluster structure, which is especially suitable for target tracking in large-scale networks. However, the static cluster membership prevents sensors in different clusters from collaborating and sharing information with each other, which causes the boundary problem when the target moves across or along the boundaries of clusters. In this paper, HCTT solves the boundary problem by integrating on-demand dynamic clustering into the scalable cluster structure.

3. System Model and Problem Statement

In this section, we first present the system model, and then we describe the boundary problem for target tracking in cluster-based WSNs.

3.1. System Model. We assume that a large-scale WSN consisting of n static sensor nodes is deployed in a two-dimensional area of interest for detecting a single moving target. The sink node is deployed at the center of the network. We assume that sensor nodes are randomly deployed following a Poisson distribution with a density of λ. That is, given an area A, the number of sensor nodes in the area, $N(A)$, follows a Poisson distribution with parameter λA as follows:

$$Pr\left(N\left(A\right)=k\right)=\frac{\left(\lambda A\right)^{k}}{k!}\cdot\left(e^{-\lambda A}\right),\quad k=0,1,\ldots,\infty. \tag{1}$$

We assume that each sensor node has an identical communication range r_c. The sensor nodes within the communication range of a sensor node are called its neighbor nodes. That is, $N(v_i)=\{v_j \mid d(l_i,l_j)\le r_c\}$ is the set of neighbor nodes of sensor node v_i, where l_i is the location of v_i, l_j is the location of v_j, and $d(l_i,l_j)$ is the Euclidean distance between v_i and v_j.

In general, the received signal of a sensor node about the target reduces with the increase of the distance between the sensor node and the target. An attenuated disk sensing model is adopted to capture the sensing quality as follows:

$$r_i=\begin{cases}\dfrac{\beta}{d^{\alpha}\left(l_i,L\right)}, & d\left(l_i,L\right)\le r_s,\\ 0, & \left(l_i,L\right)>r_s,\end{cases} \tag{2}$$

where r_i is the received signal of sensor node v_i, β is the original strength of emitted signal of the target, α is the path attenuation exponent, r_s is the sensing range, l_i is the location of v_i, L is the target location, and $d(l_i,L)$ is the Euclidean distance between sensor node v_i and the target.

Note that the sensing region of a sensor node v_i, denoted by $R(v_i,r_s)$, is a disk with center l_i and radius r_s. A target will be detected by the sensor v_i when it appears in the sensing region $R(v_i,r_s)$. Conversely, only sensor nodes within the distance of r_s of the target can detect the target. Therefore, we call the disk centered at the target and with radius r_s as the *monitoring region* of the target, as any sensor within this region can monitor the target.

Each node can operate in three states. It can transmit packets, receive packets, and sense the target in active state. In sensing state, it can only perform sensing operation. In sleep state, it sleeps for most of time and wakes up periodically to sense the target and listen to messages. Since communication operation dominates the energy consumption, we mainly consider the energy consumption for communication in this paper.

We assume that the WSN is organized as m clusters by using any suitable clustering algorithm in [28]. In this paper, we use the LEACH [14] to organize nodes into static clusters. The LEACH protocol is an energy efficient adaptive clustering protocol, which is distributed and energy balanced. Each cluster i has n_i nodes including one cluster head and n_i-1 1-hop members. Each node is aware of its own location and some local information of its neighbors but has no global topology information. To fulfill the monitoring task, cluster heads are responsible for selecting some members to monitor the target and deal with the handoff process when the target moves.

To ease the description of the protocol, we adopt the following notations throughout the paper:

(i) n: the number of sensor nodes

(ii) m: the number of disjoint static clusters

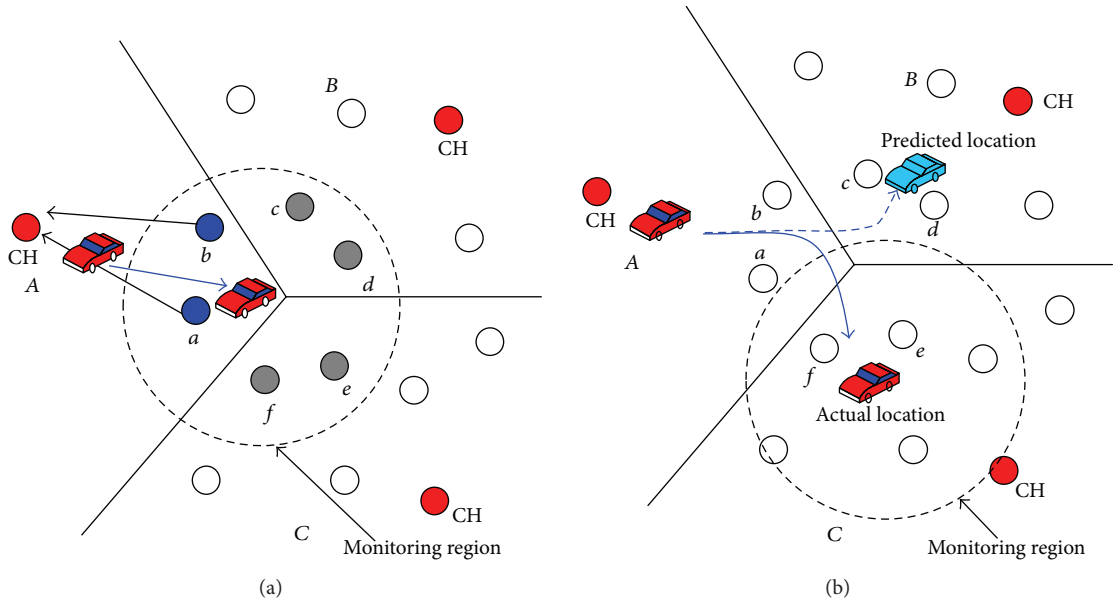

Figure 2: The boundary problem in a cluster-based WSN: (a) high localization uncertainty due to insufficient active nodes, (b) loss of target due to incorrect prediction of the target location.

(iii) r_c: the communication range of each node

(iv) r_s: the sensing range of each node

(v) $R(v_i, r_s)$: the sensing region of node v_i with sensing radius r_s

(vi) l_i: the location of node v_i. $l_i = (x_i, y_i)$ where (x_i, y_i) are the coordinates of the node.

(vii) $L(t)$: the location of the target at time t

(viii) G: the wireless sensor network. $G = (V, E)$

(ix) V: the set of all sensor nodes. $V = \{v_1, v_2, \ldots, v_n\}$

(x) E: the set of edges referred to all connected links of nodes. $E = \{e = (v_i, v_j) \mid \{v_i, v_j\} \subseteq V \wedge d(l_i, l_j) \leq r_c \wedge i \neq j\}$

(xi) $N(v_i)$: the neighbor set of node v_i. $N(v_i) = \{v_j \mid d(l_i, l_j) \leq r_c\}$

(xii) C_i: the set of nodes in cluster i. $C_i \subseteq V$. Note that $\bigcup_{i=1}^{m} C_i = V$ and $C_i \bigcap C_j = \phi$, where $\{i \neq j \mid i, j = 1, 2, \ldots, m\}$

(xiii) $C(v_i)$: the cluster which node v_i belongs to

(xiv) H: the set of all cluster heads. $H = \{v_{H_1}, v_{H_2}, \ldots, v_{H_m}\} \subset V$, where v_{H_i} denotes the cluster head of cluster i.

3.2. Boundary Problem. When tracking a target in a surveillance area, multiple nodes surrounding the target collaborate to make the collected information more complete, reliable, and accurate. There is no problem when the target is inside a cluster, as all activated sensors belong to the same cluster, and they can communicate effectively. However, when the target moves across or along the boundaries of multiple clusters, the boundary problem occurs. That is, the local node collaboration becomes incomplete and unreliable because sensor nodes that can monitor the target belong to different clusters, which increases the uncertainty of the localization of the target or even results in the loss of the target due to the insufficient sensing reports or the incorrect prediction of the target's location.

Figure 2 shows two cases of the boundary problem for target tracking in a cluster-based WSN. As the target is inside cluster A, cluster A is activated to track the target, whereas clusters B and C are in the sleep state for energy saving purpose. For the case in Figure 2(a), when the target moves close to the boundaries of clusters, nodes a, b, c, d, e, and f, which can monitor the target, belong to three different clusters A, B, and C. Only nodes a and b can sense the target, as they are active, while other nodes cannot, as they are in the sleep state. The network cannot successfully locate the target due to insufficient information, which may affect the accuracy of predicting the target's next position or even result in the loss of the target. For the case shown in Figure 2(b), the next location of the target is predicted to be in cluster B whereas the target actually moves into cluster C. Nodes in cluster B, are activated in advance according to the predicted location of the target. However, none of them can sense the target since the target moves into cluster C. Therefore, prediction error may also result in the loss of the target.

In this paper, we propose a hybrid cluster-based target tracking protocol (HCTT) for efficient target tracking in a cluster-based WSN. HCTT integrates on-demand dynamic clustering into a scalable cluster-based structure with the help of boundary nodes. That is, when the target moves inside a cluster, the static cluster is responsible for local node collaboration and target tracking; when the target approaches the boundaries of clusters, a dynamic clustering process will be triggered to form a dynamic cluster to solve the boundary

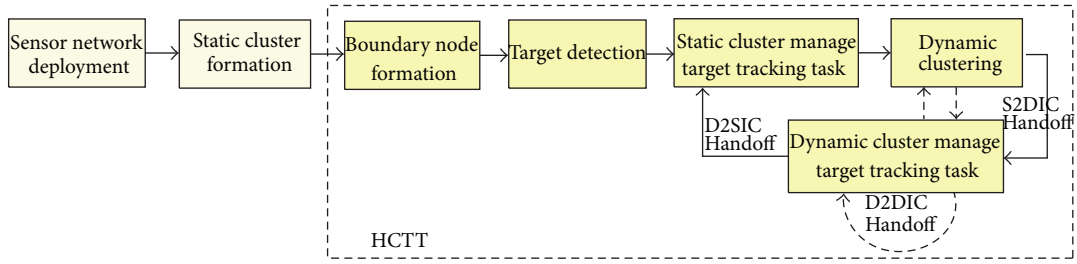

Figure 3: Overview of hybrid cluster-based target tracking protocol.

problem. The on-demand dynamic cluster will disappear soon after the target moves away from the boundaries. As the target moves in the network, static clusters and on-demand dynamic clusters alternately handle the tracking task effectively.

4. Hybrid Cluster-Based Target Tracking Protocol (HCTT)

In this section, we first give a high-level overview of HCTT and then describe the protocol in detail. Figure 3 outlines the overview of the system and illustrates the flowchart of HCTT. After being deployed, sensor nodes are organized into static clusters according to any suitable clustering algorithm. Boundary nodes of each cluster are also formed. When a target is in the network, a static cluster sensing the target wakes up to track the target. When the target approaches the boundary, the boundary nodes can detect the target, and an on-demand dynamic cluster will be constructed in advance for smoothly tracking the target before the target reaches the boundary. As the target moves across the boundaries, static clusters and on-demand dynamic clusters alternately manage the tracking task. Different types of intercluster handoff (i.e., the S2DIC handoff from a static cluster to a dynamic cluster, the D2SIC handoff from a dynamic cluster to a static cluster, and the D2DIC handoff from a former dynamic cluster to a new dynamic cluster) take place between any two sequential clusters. Moreover, the dynamic cluster will dismiss after the target moves away from the boundary and enters another cluster. With the help of boundary nodes, HCTT integrates on-demand dynamic clustering into a scalable cluster-based WSN for target tracking, which facilitates the sensors' collaboration among clusters and solves the boundary problem.

In the following, we elaborate on the design and implementation of three major components of HCTT, including the boundary node formation, dynamic clustering, and intercluster handoff.

4.1. Boundary Node Formation. In HCTT, a dynamic cluster will be constructed when the target approaches the boundaries of multiple clusters. A challenging issue is how the system finds the scenario when the target is approaching the boundaries, especially in a fully distributed way. As sensor nodes are randomly deployed, static clusters are usually formed irregularly. It is difficult or even impossible to calculate the geometrical boundaries of irregular clusters. In this paper, we use *boundary nodes* to solve this issue in a fully distributed way.

Definition 1. Boundary node of a static cluster: A node v_i is defined as a boundary node of its static cluster if there exists at least one of its neighbor nodes v_j, such that $d(l_i, l_j) \leq r_s$ and $C(v_i) \neq C(v_j)$.

In contrast, a node is defined as an *internal node* of its static cluster if it is not a boundary node. As each node is aware of its own location and its neighbor information, it checks its neighbor list to determine whether there exists a node belonging to another cluster within its sensing range. If yes, it is a boundary node; otherwise, it is an internal node. The *boundary node set* of a cluster C_i, denoted by $B(C_i)$, is formed by all the boundary nodes in C_i. The *internal node set* of a cluster C_i, denoted by $I(C_i)$, is formed by all the internal nodes in C_i. The pseudocodes of boundary node formation process are described in Algorithm 1.

After boundary nodes are identified, each cluster can be partitioned into three parts: safety region, boundary region, and alert region.

Safety Region. The *safety region* of a cluster C_i, denoted by $R_S(C_i)$, is the region in cluster C_i that can be monitored by at least one internal node of C_i, but not by any boundary node of C_i. That is, $R_S(C_i)$ can be formulated as

$$R_S(C_i) = \bigcup_{\forall v_i \in I(C_i)} R(v_i, r_s) - \bigcup_{\forall v_i \in B(C_i)} R(v_i, r_s). \quad (3)$$

Boundary Region. The *boundary region* of a cluster C_i, denoted by $R_B(C_i)$, is the region that can be monitored by some boundary nodes of both C_i and any of its adjacent clusters at the same time. That is, $R_B(C_i)$ can be formulated as

$$R_B(C_i) = \bigcup_{\forall v_i \in B(C_i), \forall v_j \in B(C_j), \forall C_j \neq C_i} \left(R(v_i, r_s) \cap R(v_j, r_s)\right). \quad (4)$$

Alert Region. The *alert region* of a cluster C_i, denoted by $R_A(C_i)$, is the region that can be monitored by at least one

Input: Graph $G = \{V, E\}$, cluster sets C_i, $i = 1, 2, \ldots, m$.
Output: $B(C_i)$, $i = 1, 2, \ldots, m$
(1) **for** each cluster C_i **do**
(2) $B(C_i) \leftarrow \phi$
(3) **for** each node $v_j \in C_i$ **do**
(4) **while** $\exists v_k \in N(v_j)$ such that $d(l_k, l_j) \leq r_s$ and $C(v_j) \neq C(v_k)$ **do**
(5) $v_j \cdot state \leftarrow boundary$
(6) $B(C_i) \leftarrow B(C_i) \cup \{v_j\}$

Algorithm 1: Boundary node formation.

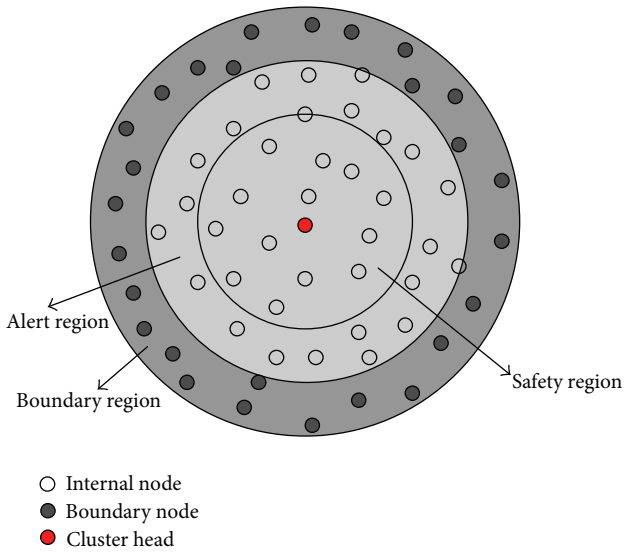

Figure 4: Demonstration of safety region, alert region, and boundary region for a circular cluster.

boundary node of C_i but not belongs to the boundary region of C_i. That is, $R_A(C_i)$ can be formulated as

$$R_A(C_i) = \bigcup_{\forall v_i \in B(C_i)} R(v_i, r_s) - R_B(C_i). \quad (5)$$

Figure 4 illustrates the three different regions for a circular cluster. The white region denotes the safety region, the light gray region denotes the alert region, and the dark gray region denotes the boundary region. We can observe that if the target moves from the safety region into the boundary region through the alert region, nodes belonging to different clusters will detect the target. More specifically, we have the following lemmas.

Lemma 2. *All boundary nodes lie in the boundary region.*

Proof. For any boundary node v_i of cluster C_i, by definition, there exists a neighbor node v_j of cluster C_j, such that $d(l_i, l_j) \leq r_s$ and $C(v_i) \neq C(v_j)$. That is, v_j is also a boundary node of cluster C_j. Since $d(l_i, l_j) \leq r_s$, we can get v_i lies in $R(v_i, r_s) \cap R(v_j, r_s)$. □

Lemma 3. *If nodes detecting the target are all internal nodes of a cluster the target moves into the safety region of the cluster, and vice versa.*

Proof. If nodes detecting the target are all internal nodes of a cluster, which suggests that the target cannot be detected by any boundary node. According to (3), the target moves into the safety region. It is also true that if the target moves into the safety region of a cluster, obviously all nodes detecting the target are internal nodes of the cluster. □

Lemma 4. *If nodes detecting the target belong to different clusters, the target moves into the boundary region of one cluster and vice versa.*

Lemma 5. *If nodes detecting the target all belong to one cluster and there is at least one boundary node among them, the target moves into the alert region of the cluster and vice versa.*

The proofs of Lemmas 4 and 5 are similar to that of Lemma 3.

Based on Lemmas 3, 4, and 5, a cluster head can decide which region the target locates when it receives data reports from sensor nodes.

4.2. Dynamic Clustering. Dynamic clustering is triggered on demand in order to solve the boundary problem in a cluster-based WSN. There are two cases in which a dynamic cluster is needed. The first case is shown in Figure 5(a) where the current active cluster is cluster A. When the target moves from point a to point c via point b, a dynamic cluster $D1$ will be constructed for smoothly tracking the target. The second case is shown in the Figure 5(b) where the current active cluster is a dynamic cluster $D2$. When the target moves along the boundaries from point m to point q via points o and p, using only one dynamic cluster cannot satisfy the target tracking requirement. In this case, another dynamic cluster $D3$ will be constructed when the target moves from point o to point p. Note that clusters of square shapes in Figure 5 are just used to help explain the handoff problem in target tracking. The clusters might not have any regular shape in practice.

Though the dynamic cluster must be constructed in advance before the target moves across the boundaries of clusters, it is costly if the dynamic cluster is constructed too early, which not only wastes nodes' energy consumption but

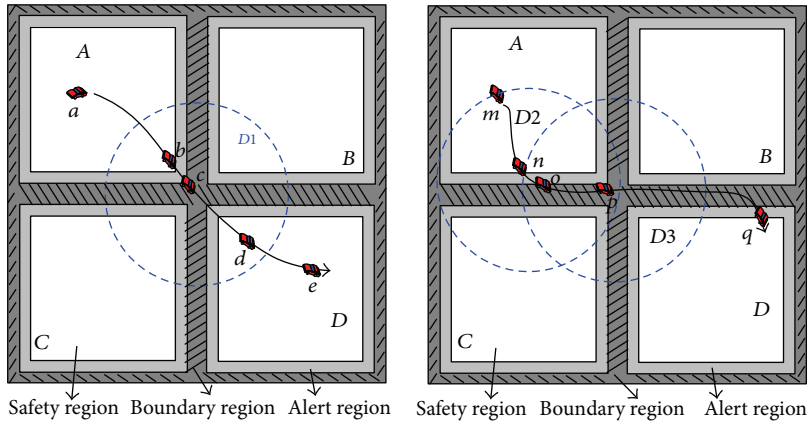

Figure 5: Dynamic clustering and handoff between clusters: (a) one dynamic cluster, (b) two consecutive dynamic clusters.

also may become useless quickly. As stated in Lemma 5, if the target is detected by any boundary node, the target moves into the alert region, which means that the target is closely approaching the boundary region. It is reasonable to trigger the on-demand dynamic clustering process when the target moves into the alert region, that is, when there is at least one boundary node detecting the target. The dynamic clustering process consists of three phases: leader selection, dynamic cluster construction, and boundary node formation.

4.2.1. Leader Selection. The first step to construct a dynamic cluster is to elect a cluster head from sensor nodes. The dynamic clustering process should be triggered once the target moves into the boundary region. Therefore, the first boundary node detecting the target is supposed to be the cluster head and construct a dynamic cluster. However, sensor nodes may detect the target at the same time, so these nodes may compete to be the cluster head. Therefore, an algorithm must be proposed to guarantee only one boundary node to be the cluster head.

Once a boundary node detects a target, if it does not receive any *election* message from its neighbor nodes, it broadcasts an *election* message to its one-hop neighbor nodes. If a boundary node receives some *election* messages before detecting the target, it would not broadcast any message. The *election* message includes the ID, the received signal of the target, and the residual energy of the sensor node. After it broadcasts the *election* message, it waits for a predefined timeout period to receive the *election* message from other sensor nodes. A sensor node may or may not receive the *election* message during the timeout period; otherwise, if it does not receive any *election* message after the timeout period, it will elect itself as the cluster head. If it receives multiple *election* messages from its neighbors, the node with the highest received signal will be elected as the cluster head. In case some nodes have the same received signal, the residual energy can be used to break the tie.

4.2.2. Dynamic Cluster Construction. After a node is elected as the cluster, it broadcasts a *recruit* message asking its neighbor nodes to join in the cluster. Each node receiving the *recruit* message establishes a new table containing the information of the dynamic cluster, for example, the ID of the leader and the working state of the dynamic cluster, and then replies a *confirm* message to the leader. Upon receiving the *confirm* message from a node, the leader puts the node into its member list.

Note that once a dynamic cluster is constructed, a node may belong to a static cluster and a dynamic cluster at the same time. When the node generates some data, it needs to decide which cluster head it should report to. This is important for the inter-cluster handoff procedure, and we will describe the details in Section 4.3.

4.2.3. Boundary Node Formation. Boundary nodes are also necessary for dynamic clusters. It can inform the cluster head of the dynamic cluster in advance if the target is about to leave the dynamic cluster.

Definition 6. Boundary node of a dynamic cluster. A node v_i is defined as a boundary node of a dynamic cluster D if there exists at least one neighbor node v_j, such that $v_j \notin D$ and $d(l_i, l_j) \leq r_s$.

The way of forming boundary nodes of a dynamic cluster is different from that of a static cluster. Each node v_i checks its neighbor list to see if there exists a neighbor node that does not belong to the dynamic cluster and also is within its sensing range. If such a neighbor node can be found, v_i is a boundary node of the dynamic cluster; otherwise, it is an internal node of the dynamic cluster. Once boundary nodes of the dynamic cluster are identified, the dynamic clustering process is completely finished. After the dynamic cluster is constructed, it stays in the waiting state for the coming of the target. The pseudocodes of the dynamic clustering process are described in Algorithm 2.

Phase I: Leader selection
(1) Leader node $L_{dc} \leftarrow \phi$
(2) **for** each sensor node detecting the target **do**
(3) **if** it does not receive any *election* message **then**
(4) broadcasts its *election* message and waits for a predefined timeout period for election messages from neighbor nodes
(5) **if** it does not receive any message during the period **then**
(6) elects itself as the cluster head L_{dc}
(7) **else**
(8) elects the node with the highest received signal as the cluster head L_{dc}
Phase II: Dynamic cluster construction
(1) dynamic cluster set $C_{dc} \leftarrow \phi$
(2) L_{dc} broadcasts a *recruit* message
(3) **while** node v_k receiving the recruit message **do**
(4) v_k.leader $\leftarrow L_{dc}$
(5) v_k replies a *confirm* message
(6) **while** L_{dc} receives a *confirm* message from v_k **do**
(7) $C_{dc} \leftarrow C_{dc} \cup \{v_k\}$
Phase III: Boundary node formation
(1) boundary nodes set $B_{dc} \leftarrow \phi$
(2) **for** each node $v_k \in C_{dc}$ **do**
(3) **while** $\exists v_p \in N(v_k)$ such that $d(l_k, l_p) \leq r_s$ and $v_p.leader \neq L_{dc}$ **do**
(4) v_k.state $\leftarrow$ boundary
(5) $B_{dc} \leftarrow B_{dc} \cup \{v_k\}$

Algorithm 2: Dynamic clustering.

Table 1: Message reporting priority.

S	D	Reported cluster head
Active	Idle	CH_s
Sleep	Idle	CH_d
Sleep	Active	CH_d

4.3. Intercluster Handoff. Before introducing the details of inter-cluster handoff process, we first give the message reporting priority order used in our protocol.

Message Reporting Priority. Each cluster can work at one of three states: active, idle (waiting), and sleep. The active state has the highest priority for message reporting, while the sleep state has the lowest priority. If a node belongs to a static cluster and a dynamic cluster simultaneously, it always reports to the cluster head with higher priority. One exception case is that, if the node is a boundary node of a static cluster, it reports to the headers of both clusters. Table 1 illustrates different data reporting scenarios where a node belongs to a static cluster and a dynamic cluster at the same time. Here, S and D denote a static cluster and a dynamic cluster, respectively, and CH_s and CH_d denote the cluster heads of S and D, respectively.

Generally, the tracking task should be handled by the most suitable cluster. As the target moves in the network, the task should be handed over from the current cluster to another more suitable one, which is taken charge of by the inter-cluster handoff process. The inter-cluster handoff occurs in three typical scenarios: handoff from a static cluster to a dynamic cluster (e.g., from cluster A to cluster $D1$ as shown in Figure 5(a)), handoff from a dynamic cluster to a static cluster (e.g., from $D1$ to D as shown in Figure 5(a)), and handoff from a dynamic cluster to another dynamic cluster (e.g., from $D2$ to $D3$ as shown in Figure 5(b)). In the following part, we will describe how to efficiently implement these three types of handoff processes.

4.3.1. S2DIC Handoff. When the target locates within a cluster, it is tracked by the current active static cluster. As the target moves into the alert region, boundary nodes can detect the target. Upon detecting the target by a boundary node, a dynamic cluster is constructed in advance for the coming of the target. However, the handoff should not take place right away once the new dynamic cluster is constructed, since the movement of the target is uncertain. The target may move towards the boundary, but it may also turn around and move away from the boundary. As shown in Figure 6, when the target moves to point b in the alert region, a dynamic cluster $D1$ is constructed before the target moves into the boundary region. If the target's trajectory is T1, the dynamic cluster is more suitable than the active static cluster for tracking the target. However, the target may follow trajectories T2 or T3. For these trajectories, it is inappropriate to perform the handoff, since the target does not actually move into the boundary region. Especially for the trajectory of T3,

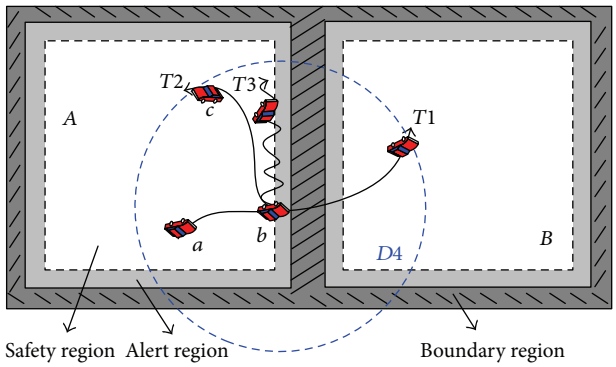

FIGURE 6: Different moving trajectories of the target.

unnecessary dynamic clustering and handoff may take place frequently if we do not have an effective handoff mechanism.

To minimize unnecessary dynamic clustering and handoff, the S2DIC handoff starts when the target is confirmed to be in the boundary region. When the dynamic cluster is constructed, some boundary nodes of the current active static cluster join into the dynamic cluster. These nodes are in the waiting state, while they keep detecting the target. When any of these nodes detects the target, it sends the sensing report to the cluster head of the dynamic cluster. Once this cluster head receives the sensing report, the target is confirmed to be in the boundary region, and the handoff process starts.

The S2DIC handoff procedure is shown in Figure 7(a): the dynamic cluster head, say L_{dc}, first sends a *request* message to the active static cluster head CH_i to ask for the handoff of the leadership. CH_i replies a *handoff* message containing the historical estimations of the target's locations and hands the leadership over to L_{dc}. Once L_{dc} receives the *handoff* message, it first broadcasts a *work* message to activate all its member nodes and then replies an *acknowledge* message to CH_i. After receiving the *acknowledge* message, CH_i broadcasts a *sleep* message to its member nodes. Each member node not belonging to the active dynamic cluster goes into the sleep state to save energy consumption.

If the dynamic cluster is not suitable for tracking the target, it will be dismissed. As shown in Figure 6, when the target moves to point c following the trajectory $T2$, the dynamic cluster $D4$ is not effective anymore and will be dismissed. The dynamic cluster dismissal procedure works as follows: for a dynamic cluster in idle state, if the cluster head of the dynamic cluster receives sensing reports from the boundary nodes of the dynamic cluster, it confirms that the dynamic cluster is not needed anymore. Then the cluster head of the dynamic cluster sends a *resign* message to inform the cluster head of the active static cluster. After getting the acknowledged message from the cluster head of the active static cluster, it broadcasts a *dismiss* message to all its members. Each node, when receiving the *dismiss* message, quits the dynamic cluster and deletes the information table built for the dynamic cluster.

4.3.2. D2SIC Handoff. When a dynamic cluster is currently handling the tracking task, nodes sensing the target send their sensing reports to the cluster head of active dynamic cluster. If none of these sensing nodes is a boundary node of any static cluster, the target has moved away from the boundary region and entered the safety region of a static cluster, then the D2SIC handoff process is triggered.

The procedure of D2SIC handoff is shown in Figure 7(b): the active cluster head L_{dc} sends a *handoff* message to the cluster head of next static cluster CH_j. Once CH_j receives a *handoff* message, it broadcasts a *work* message to wake up its member nodes, then replies an *acknowledge* message to L_{dc}. After receiving the *acknowledge* message, L_{dc} broadcasts a *dismiss* message. Each node, when receiving the *dismiss* message, quits the dynamic cluster and deletes the information table for the dynamic cluster.

4.3.3. D2DIC Handoff. When a dynamic cluster is currently active for the tracking task, the handoff condition for the D2SIC handoff may not be met as the target moves into the boundary region. As shown in Figure 5(b), one dynamic cluster $D2$ cannot track the target all the time when the target moves along the boundaries of static clusters. In this scenario, the D2DIC handoff can construct another new dynamic cluster $D3$ to continue the tracking task if some boundary nodes of the active dynamic cluster detect the target. Note that the D2SIC handoff has a higher priority than the D2DIC handoff, as the latter one will cause more cost and longer delay than the former one.

The new dynamic cluster is, generally, a more preferable choice than the previous one for tracking the target, as the target is at the center of the new dynamic cluster, while it is at the boundary of the previous dynamic cluster. Therefore, the handoff takes place immediately after the new dynamic cluster is constructed. The D2DIC handoff procedure is shown in Figure 7(c): the new cluster head $L_{dc'}$ sends a *request* message to ask for the handoff of the leadership. The current active cluster head L_{dc} replies a *handoff* message containing the historical estimations of the target's location to hand over the leadership to the cluster head of the new dynamic cluster. After receiving the leadership, $L_{dc'}$ broadcasts a *work* message to inform its members to be responsible for tracking the target. Each node detecting the target sends the sensing reports to $L_{dc'}$. Then $L_{dc'}$ replies an *acknowledge* message to the L_{dc}. After receiving the *acknowledge* message, L_{dc} broadcasts a *dismiss* message. Each node, when receiving the *dismiss* message, quits from the dynamic cluster and deletes the information table for the dynamic cluster.

5. Analysis of HCTT

In this section, we will present the computation complexity and communication complexity for proposed HCTT.

As mentioned before, we have assumed that n static sensor nodes are randomly deployed in the area of interest. The deployment of these nodes follows the Poisson distribution with density of λ. Therefore, the average number of neighbor nodes of one node is $\lambda\pi r_c^2 - 1$.

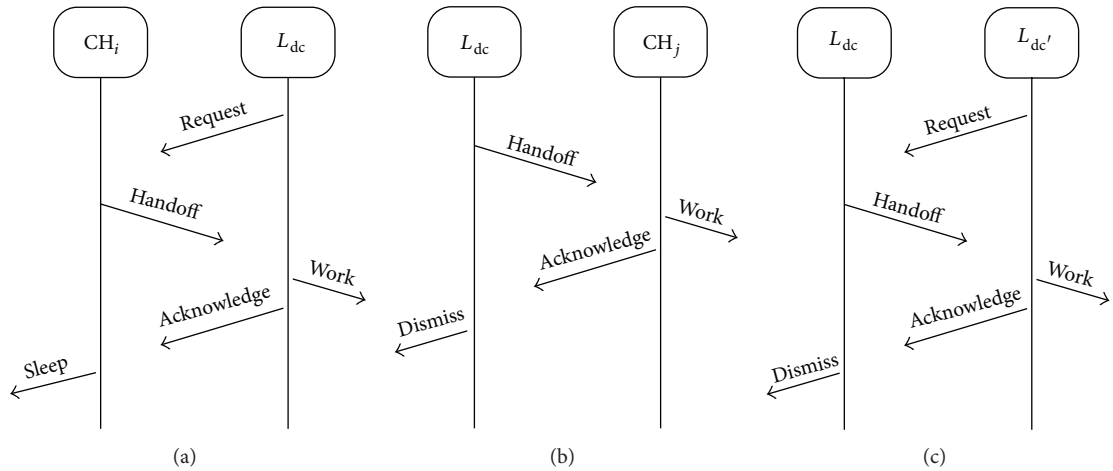

Figure 7: Illustration of intercluster handoff processes: (a) S2DIC handoff process, (b) D2SIC handoff process, and (c) D2DIC handoff process.

Theorem 7. *The computation complexity and communication complexity of the boundary node formation process are both* $O(n)$.

Proof. In the boundary node formation process, each node broadcasts a message to its one-hop neighbors, so the number of messages transmitted is proportional to the number of nodes. Therefore, the communication complexity of the boundary node formation process is $O(n)$. As presented in Algorithm 1, each node computes to judge whether it is a boundary node or not. In the worst case, each node needs to check all its neighbors to make the decision; that is, $\lambda\pi r_c^2 - 1$ times are computed to judge whether it is a boundary node or not. For the whole network, $n * (\lambda\pi r_c^2 - 1)$ times are computed in the worst case to complete the process of boundary node formation. The density λ and communication range r_c are usually fixed after deployment, so $\lambda\pi r_c^2 - 1$ can be considered as a constant. Therefore, the computation complexity of the process of boundary node formation is also $O(n)$. □

Theorem 8. *The computation complexity and communication complexity of the dynamic clustering process are both* $\Theta(1)$.

Proof. Dynamic clustering process consists of three phases: leader selection, dynamic cluster construction, and boundary node formation. As presented in Algorithm 2, all the messages transmitted are constrained in a one-hop cluster; that is, the number of messages transmitted is determined by λ and r_c, which can be considered as a constant. Therefore, the communication complexity is $\Theta(1)$. In the worst case, each dynamic node needs to check all its neighbors to decide whether it is a boundary node of the dynamic cluster or not. Since there are $\lambda\pi r_c^2$ nodes for each dynamic cluster, the overall computation overhead in the worst case is $(\lambda\pi r_c^2) * (\lambda\pi r_c^2 - 1)$, which also can be considered as a constant. Therefore, the computation complexity of the dynamic clustering process is $\Theta(1)$. We can conclude that the dynamic clustering is a local process, which is independent to the size of network. □

Theorem 9. *The computation complexity and communication complexity of the inter-cluster handoff process are both* $\Theta(1)$.

Proof. Inter-cluster handoff includes three schemes: S2DIC handoff, D2SIC handoff, and D2DIC handoff. Figure 7 shows that inter-cluster handoff is a local process. The message overhead is much smaller than $\lambda\pi r_c^2$, since the process only requires some control packets. During the handoff process, each node involved makes its decision locally to be active or go to sleep. In the worst case, sensor nodes of two clusters are involved. That is, the computation overhead in the worst case is $2\lambda\pi r_c^2$, which can be considered as a constant. Therefore, the communication complexity and computation complexity of the inter-cluster handoff process are both $\Theta(1)$. We can conclude that the inter-cluster handoff is a local process, which is independent to the size of network. □

Theorem 10. *The computation complexity and communication complexity of the hybrid cluster-based target tracking are both* $O(n)$.

Proof. Hybrid cluster-based target tracking protocol consists of three major components: boundary node formation, dynamic clustering, and inter-cluster handoff. Since the computation complexity and message complexity of the boundary node formation process are $O(n)$, the computation complexity and message complexity of HCTT are also $O(n)$. □

Although the computation complexity and communication complexity of HCTT are $O(n)$, we can see that HCTT is insensitive to the size of network if we overlook the cost of boundary node formation process. We have mentioned that, as described in [14], the static cluster structure will not change until a new round of clustering process. The period between two sequential rounds of clustering processes is usually quite long. The process of boundary node formation only takes place once for one static cluster structure. Thus, during each period of two sequential rounds, only dynamic clustering and inter-cluster handoff processes take place as targets move,

of which the computation complexity and communication complexity are both $\Theta(1)$. Therefore, in certain sense, HCTT is scalable, as the size of network increases.

6. Performance Evaluation

In this section, we evaluate the performance of our proposed scheme through simulations. Since our HCTT scheme is proposed to solve the boundary problem in a cluster-based WSN, we mainly compare the performance of HCTT with that of DPT [11]. Moreover, we further compare the performance of HCTT with other two typical tracking protocols, DCTC [7] and ADCT [6].

We use MATLAB to perform our simulations. The simulation environment is configured as follows. A total of 6000 sensor nodes are randomly deployed in the area of $400 \times 400\,\text{m}^2$ for monitoring a moving target. Sensor nodes are organized as static clusters according to the LEACH protocol. The sensing range of each node, r_s, is fixed at 10 m. The transmission range of each node, r_c, is set to be at least twice as large as the sensing range, that is, $r_c/r_s \geq 2$. The size of a control message used for constructing dynamic clusters is 10 bytes. The size of a sensing report generated by each node for detecting the target is 40 bytes. The movement of a target follows the random-way point model. The target randomly chooses a destination and moves toward this destination at a random speed in [5, 10] m/s. On arriving at the destination, the node pauses for a predetermined time and then repeats the process. The prediction error, which refers to the distance difference between the estimated location computed by the system and the real location of the target, varies from 0 to r_s. Moreover, each sensor node adopts the energy consumption model presented in [14]. To transmit a k-bit packet with a range r_c, the consumed energy is $E = E_e \cdot k + \epsilon \cdot k \cdot r_c^2$. Typically, $E_e = 50\,\text{nJ/bit}$ and $\epsilon = 10\,\text{pJ/bit/m}^2$. For all the simulation results presented in this paper, each data point is an average of 100 experiments.

We use the following metrics to evaluate the performance of our proposed scheme:

(i) number of dynamic clusters: the number of dynamic clusters generated during the target tracking process;

(ii) missing probability: the probability of missing the target as the target moves in the network;

(iii) sensing coverage: the ratio of the number of nodes sensing the target to the number of nodes within the monitoring region;

(iv) energy consumption: the energy consumed for transmitting control messages and sensing reports.

Note that, although different localization approaches affect the performance of target tracking significantly, they are not the major concern of this paper. Here, we use the metric of the sensing coverage to indicate the accuracy level of location estimates of the target, since the accuracy of localization is related to the information collected from nodes surrounding the target. It is expected that all nodes detecting the target send their sensing reports to generate reliable and accurate estimates of the target. However, many of them may not be able to sense the target successfully, as they are in the sleep state due to the prediction error or other factors. Therefore, larger sensing coverage usually means a better estimate of the target's location.

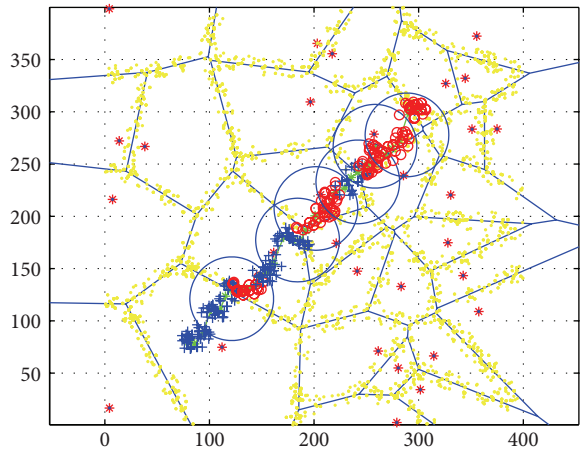

Figure 8: Snapshot of target tracking using HCTT.

Figure 8 shows a snapshot of using our proposed HCTT scheme for target tracking. Sensor nodes are organized as static clusters. Each cluster has only one cluster head, which is denoted with the snowflake mark. The yellow points denote the boundary nodes which are close to the boundaries of static clusters. The light green line denotes the moving trajectory of the target. The blue cross marks denote the nodes sensing the target in static clusters. The small red circle marks denote the nodes sensing the target in dynamic clusters, which are denoted by the large circles. We can see that dynamic clusters are triggered on-demand when the target moves to the boundaries of clusters. All these three inter-cluster handoff procedures, S2DIC handoff, D2SIC handoff, and D2DIC handoff, are shown in this snapshot when the target moves along the boundaries of clusters.

6.1. In One-Hop Static Clusters Scenario. In this scenario, sensor nodes are organized as one-hop static clusters, where each member node can send messages directly to the cluster head.

6.1.1. Number of Dynamic Clusters. Figure 9 shows the relationship between the number of dynamic clusters and the ratio of the transmission range to the sensing range r_c/r_s. We can observe that the number of dynamic clusters drops quickly, as the ratio r_c/r_s increases. That is, the smaller the ratio r_c/r_s is, the more the generated dynamic clusters are. The reason is that increasing the transmission range means increasing the size of static clusters. The larger the size of each static cluster is, the lower the probability that the target encounters the cluster's boundaries is. The number of dynamic clusters drops quickly when $r_c/r_s < 4$ and drops slowly when $r_c/r_s > 4$. Consequently, we select the turning

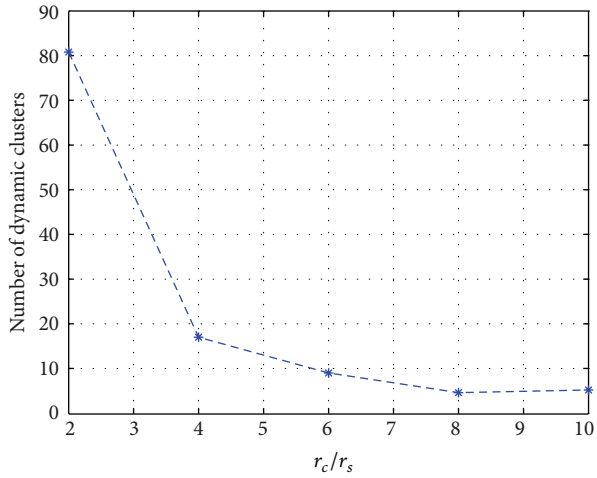

Figure 9: The number of dynamic clusters versus r_c/r_s in the HCTT algorithm.

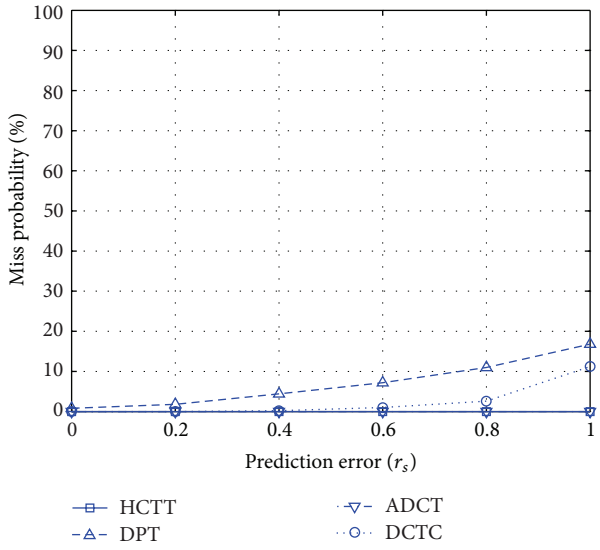

Figure 10: The missing probability versus the prediction error for all algorithms.

point of $r_c/r_s = 4$ for the performance evaluation in the following simulations.

6.1.2. Missing Probability. Figure 10 shows the effects of the prediction error on the missing probabilities of four approaches. We can see that the missing probability of DPT is much higher than any of other three ones. Even when the prediction error equals to zero, DPT still has the probability of missing the target. This shows the effects of the boundary problem in cluster-based sensor networks. Moreover, the missing probability of DPT increases, as the prediction error increases, which implies that the boundary problem will become worse when the prediction error increases. The same trend is also observed on DCTC. In comparison, the missing probabilities of ADCT and HCTT keep zero for all the time. As for ADCT, the missing probability is not affected by the

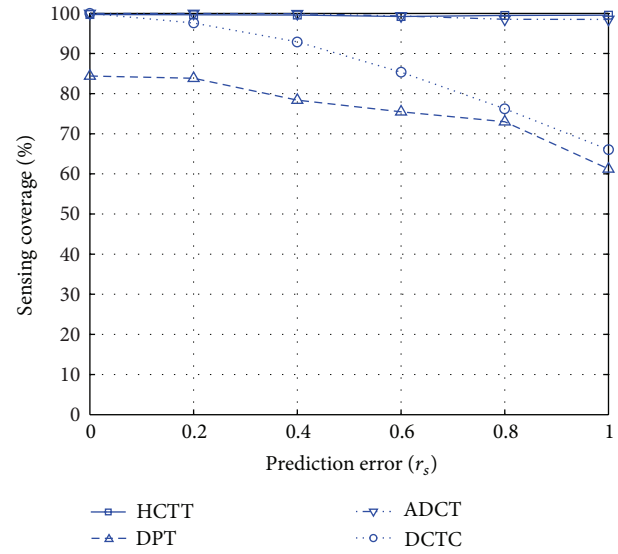

Figure 11: The sensing coverage versus the prediction error for all algorithms.

prediction error if $r_c > 2r_s$, since even in the case when the prediction error reaches r_s, all nodes in the monitoring region of the target will be waked up. HCTT does not rely on prediction algorithms to activate sensor nodes for target tracking. Therefore, the prediction error has no effect on the performance of HCTT.

6.1.3. Sensing Coverage. As shown in Figure 11, the sensing coverage of DPT and DCTC decreases when the prediction error increases, while the sensing coverage of HCTT and ADCT is not affected. It is due to the fact that increasing the prediction error results in a larger deviation of the dynamic cluster from the expected cluster, which means a large a number of nodes that should be waked up to sense the target are still staying in the sleep state. We can also see that the sensing coverage of DPT performs the worst among all approaches even when the prediction error is zero. This is the reason why DPT has the worst missing probability among all schemes. Furthermore, the sensing coverage of HCTT is nearly 100%, which means that almost all nodes surrounding the target contribute to the estimate of the target's location. Therefore, the estimate of the target's location is in general more accurate and reliable than any of other three schemes. Note that the trend of the sensing coverage in Figure 11 is opposite to the trend of the missing probability in Figure 10.

6.1.4. Energy Consumption. To evaluate the energy efficiency of HCTT, we compare the energy consumption of HCTT to other three approaches. The energy consumption presented here does not include the energy consumed for the failure recovery.

As shown in Figure 12, the energy consumptions of DPT, DCTC, and ADCT behave some drops when the prediction error increases. Since increasing the prediction error results in the drop of the sensing coverage, the energy consumption

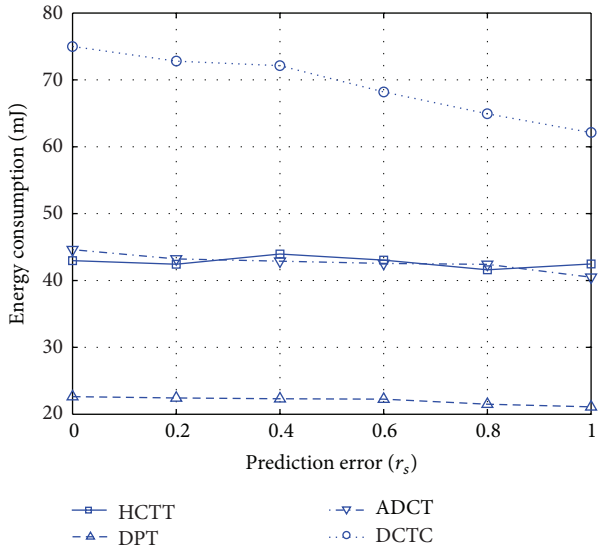

Figure 12: The Energy consumption versus the prediction error for all algorithms.

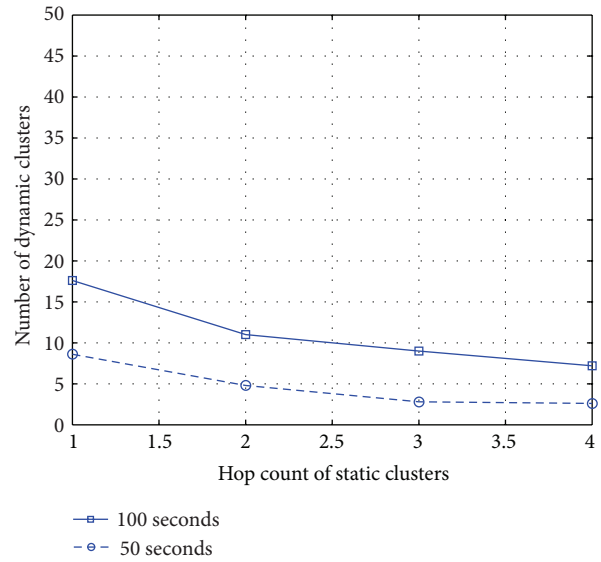

Figure 13: The number of dynamic clusters versus the hop count of static clusters in the HCTT algorithm.

drops accordingly as the number of nodes activated for sensing the target decreases. DCTC consumes more energy than other schemes, as it relies on a tree structure which incurs more overhead than a one-hop cluster structure. In contrast, DPT consumes the least energy among all schemes, as it does not need to construct and dismiss dynamic clusters. The energy consumptions of ADCT and HCTT stand between the ones of DPT and DCTC. HCTT is not the most energy efficient scheme, as there is a tradeoff between the energy consumption and missing probability/sensing coverage. Although DPT is the most energy efficient target tracking approach, it suffers the boundary problem which results in a high probability of missing the target.

Note that the performances between ADCT and HCTT are very close. However, HCTT is designed to solve the boundary problem in cluster-based networks, which implicitly takes the advantages of the cluster structure. For example, data can be easily routed from a node to the sink or another node with a low delay, which is also important for target tracking. Besides, we do not consider the energy consumption of the failure recovery of these schemes when the network loses the target. Obviously, the energy consumptions of the failure recovery of DPT, ADCT, and DCTC are much higher than that of HCTT, especially for that of DPT.

6.2. In Multihop Static Clusters Scenario. In this part, we will further evaluate the performance of HCTT in multihop cluster-based networks. Sensor nodes can be organized into multi-hop clusters via various approaches. For example, Lin and Chu [29] proposed a multi-hop clustering technique using hop distance to control the cluster structure instead of the number of nodes in a cluster. A randomized distributed multi-hop clustering algorithm for organizing the sensors into clusters is proposed in [30]. In this paper, we do not discuss how to effectively construct multi-hop cluster structure but just assume that sensor nodes are formed into multi-hop clusters. Sensor nodes are organized as grid clusters with a cluster head at the center. Each node decides the hop counts to its cluster head according to its relative distance. We use the same configurations in multi-hop cluster-based networks as those in one-hop cluster-based networks, except that the transmission range of each node is fixed to be 40 m and the prediction error is $0.4\,r_s$.

Since our proposed scheme is used to solve the boundary problem for cluster-based networks, we only compare the performance of HCTT to that of DPT in terms of hop count.

6.2.1. Number of Dynamic Clusters. Figure 13 shows the effects of the hop count of each static cluster on the number of dynamic clusters. We can see that the number of generated dynamic clusters decreases, as the hop count increases from 1 to 4. This is due to the fact that it is more frequent to encounter boundaries of static clusters if the sizes of clusters become smaller. Besides, we can see that the number of dynamic clusters increases, as the movement duration of the target increases.

6.2.2. Missing Probability. Figure 14 shows the performance of the missing probability versus the hop count of each static cluster. We can see that the missing probability of DPT drops slightly when the hop count increases from 1 to 4. This is because that it is more likely to make wrong predictions about the next working cluster when the sizes of static clusters are smaller. In comparison, the missing probability of HCTT keeps zero for all the time and does not be influenced by the hop count. This validates that HCTT can solve the boundary problem not only for one-hop cluster-based networks, but also for multi-hop cluster-based networks.

6.2.3. Sensing Coverage. As shown in Figure 15, the sensing coverage of DPT increases, as the hop count increases from 1

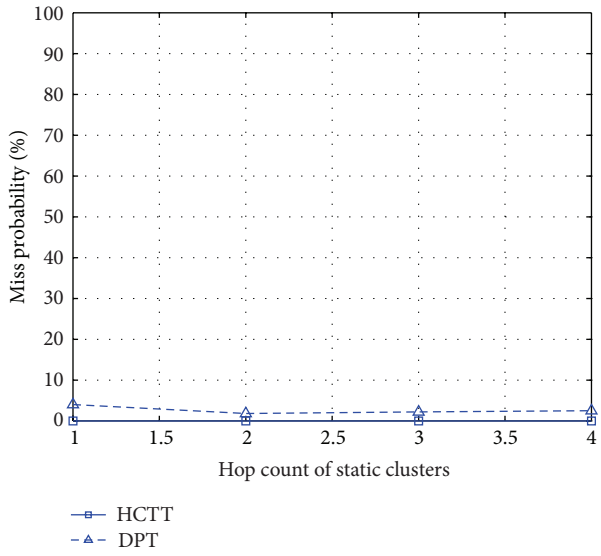

Figure 14: The missing probability versus the hop count of static clusters.

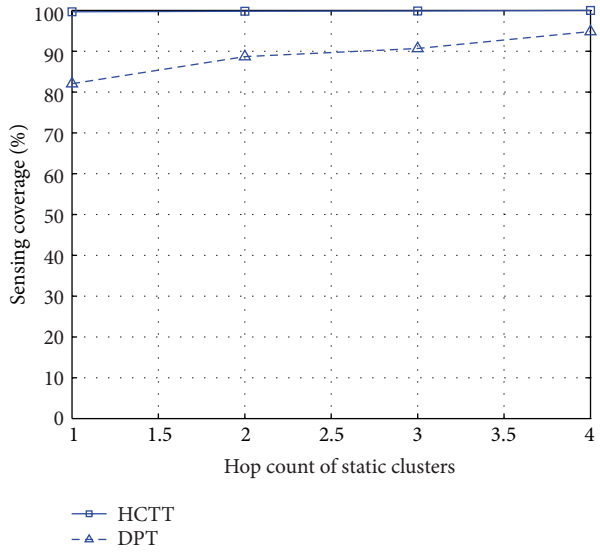

Figure 15: The sensing coverage versus the hop count of static clusters.

to 4. This explains why the missing probability drops slightly, as the hop count increases. The sensing coverage of HCTT keeps nearly 100% for all the time, which means almost all nodes sensing the target contribute to the estimate of the target's location. Consequently, we can conclude that HCTT is robust to both the prediction error and the hop count of static clusters.

6.2.4. Energy Consumption. Figure 16 shows the trend of the energy consumptions of DPT and HCTT in terms of hop count. Both the energy consumptions of DPT and HCTT largely increase, as the hop count increases from 1 to 4. This is because that each node should send its sensing data via

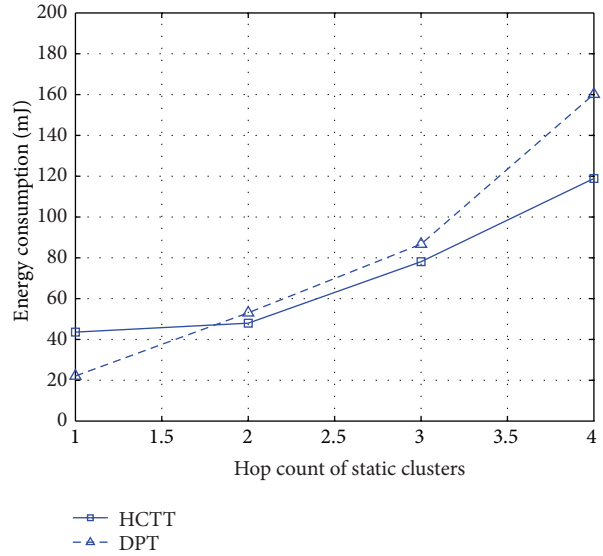

Figure 16: The Energy consumption versus the hop count of static clusters.

a multi-hop path to its cluster head in the case of multi-hop static clusters. An interesting thing is that, although the energy consumption of DPT is smaller than that of HCTT in the case of one-hop clusters, the simulation shows opposite results in the case of multi-hop clusters. Moreover, the difference between the energy consumptions of DPT and HCTT also increases, as the hop count increases. That is, the energy consumption of DPT increases faster than that of HCTT. The reason is that dynamic clusters generated at the boundaries of clusters are just one-hop clusters. The communication overhead in such one-hop dynamic clusters is much less than that in multi-hop static clusters. Overall, HCTT becomes more energy efficient than DPT in the case of multi-hop clusters.

Summary. From previous discussions, we can conclude that HCTT can solve the boundary problem not only for one-hop cluster-based networks, but also for multi-hop cluster-based networks. We can also conclude that, when the size of clusters increases, HCTT outperforms DPT in terms of missing probability, sensing coverage, and energy consumption. Furthermore, we find that HCTT is robust to both the prediction error and the hop count of static clusters for target tracking. Therefore, HCTT is an efficient mobility management approach for target tracking in cluster-based sensor networks.

7. Conclusions

Target tracking is a typical and important application of WSNs. However, tracking a moving target in cluster-based WSNs suffers the boundary problem when the target moves across or along the boundaries of clusters. In this paper, we propose a novel mobility management protocol for target tracking in cluster-based WSNs. The protocol integrates on-demand dynamic clustering into scalable cluster-based

WSNs with the help of boundary nodes, which facilitates sensors' collaboration among clusters and their smooth target tracking from one static cluster to another. The proposed algorithm solves the boundary problem of cluster-based sensor networks and achieves a well tradeoff between energy consumption and local node collaboration. The simulation results demonstrate the efficiency of the proposed protocol in both one-hop and multi-hop cluster-based sensor networks.

Acknowledgments

This work was supported in part by NSFC Grants no. 61273079 and no. 61272463, Key Lab of Industrial Control Technology Grants no. ICT1206 and no. ICT1207, Hong Kong GRF Grants PolyU-524308 and PolyU-521312, HKPU Grant A-PL84, the Fundamental Research Funds for the Central Universities no. 13CX02100A, and Zhejiang Provincial Key Lab of Intelligent Processing Research of Visual Media no. 2012008.

References

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Communications Magazine*, vol. 40, no. 8, pp. 102–105, 2002.

[2] C. R. Lin and M. Gerla, "Adaptive clustering for mobile wireless sensor networks," *IEEE Journal on Selected Areas in Communications*, vol. 15, pp. 1265–1275, 1997.

[3] M. R. Pearlman and Z. J. Haas, "Determining the optimal configuration for the zone routing protocol," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 8, pp. 1395–1414, 1999.

[4] U. C. Kozat, G. Kondylis, B. Ryu, and M. K. Marina, "Virtual dynamic backbone for mobile ad hoc networks," in *Proceedings of the International Conference on Communications (ICC '01)*, pp. 250–255, June 2000.

[5] W. P. Chen, J. C. Hou, and L. Sha, "Dynamic clustering for acoustic target tracking in wireless sensor networks," *IEEE Transactions on Mobile Computing*, vol. 3, no. 3, pp. 258–271, 2004.

[6] W. C. Yang, Z. Fu, J. H. Kim, and M. S. Park, "An adaptive dynamic cluster-based protocol for target tracking in wireless sensor networks," in *Advances in Data and Web Management*, vol. 4505 of *Lecture Notes in Computer Science*, pp. 156–167, Springer, Berlin, Germany, 2007.

[7] W. Zhang and G. Cao, "DCTC: dynamic convoy tree-based collaboration for target tracking in sensor networks," *IEEE Transactions on Wireless Communications*, vol. 3, no. 5, pp. 1689–1701, 2004.

[8] X. Ji, H. Zha, J. J. Metzner, and G. Kesidis, "Dynamic cluster structure for object detection and tracking in wireless ad-hoc sensor networks," in *Proceedings of the IEEE International Conference on Communications*, pp. 3807–3811, June 2004.

[9] G. Y. Jin, X. Y. Lu, and M. S. Park, "Dynamic clustering for object tracking in wireless sensor networks," in *Proceedings of the 3rd Iternational Conference on Ubiquitous Computing Systems (UCS '06)*, pp. 200–209, 2006.

[10] H. Medeiros, J. Park, and A. C. Kak, "Distributed object tracking using a cluster-based Kalman filter in wireless camera networks," *IEEE Journal on Selected Topics in Signal Processing*, vol. 2, no. 4, pp. 448–463, 2008.

[11] H. Yang and B. Sikdar, "A protocol for tracking mobile targets using sensor networks," in *Proceedings of the 1st IEEE International Workshop on Sensor Network Protocols and Applications*, pp. 71–81, 2003.

[12] Z. B. Wang, H. B. Li, X. F. Shen, X. C. Sun, and Z. Wang, "Tracking and predicting moving targets in hierarchical sensor networks," in *Proceedings of the IEEE International Conference on Networking, Sensing and Control*, pp. 1169–1174, 2008.

[13] Z. B. Wang, Z. Wang, H. L. Chen, J. F. Li, and H. B. Li, "Hiertrack—an energy efficient target tracking system for wireless sensor networks," in *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems*, pp. 377–378, 2011.

[14] W. B. Heinzelman, A. P. Chandrakasan, and H. Balakrishnan, "An application-specific protocol architecture for wireless microsensor networks," *IEEE Transactions on Wireless Communications*, vol. 1, no. 4, pp. 660–670, 2002.

[15] Z. B. Wang, W. Lou, Z. Wang, J. C. Ma, and H. L. Chen, "A novel mobility management scheme for target tracking in cluster-based sensor networks," in *Proceedings of the IEEE International Conference on Distributed Computing in Sensor Systems*, pp. 172–186, 2010.

[16] F. Zhao, J. Shin, and J. Reich, "Information-driven dynamic sensor collaboration for tracking applications," *IEEE Signal Processing Magazine*, vol. 19, no. 2, pp. 61–72, 2002.

[17] R. R. Brooks, C. Griffin, and D. S. Friedlander, "Self-organized distributed sensor network entity tracking," *International Journal of High Performance Computing Applications*, vol. 16, no. 3, pp. 207–219, 2002.

[18] J. Chen, K. Cao, K. Li, and Y. Sun, "Distributed sensor activation algorithm for target tracking with binary sensor networks," *Cluster Computing*, vol. 14, no. 1, pp. 55–64, 2011.

[19] F. Zhang, J. Chen, H. Li, Y. Sun, and X. M. Shen, "Distributed active sensor scheduling for target tracking in ultrasonic sensor networks," *Mobile Networks and Applications*, vol. 17, no. 5, pp. 582–593, 2011.

[20] Z. Wang, J. A. Luo, and X. P. Zhang, "A novel location-penalized maximum likelihood estimator for bearing-only target localization," *IEEE Transaction on Signal Processing*, vol. 60, no. 12, pp. 6166–6181, 2012.

[21] T. He, S. Krishnamurthy, L. Luo et al., "VigilNet: an integrated sensor network system for energy-efficient surveillance," *ACM Transactions on Sensor Networks*, vol. 2, no. 1, pp. 1–38, 2006.

[22] T. He, P. Vicaire, T. Yant et al., "Achieving real-time target tracking using wireless sensor networks," in *Proceedings of the 12th IEEE Real-Time and Embedded Technology and Applications Symposium*, pp. 37–48, April 2006.

[23] P. Cheng, J. M. Chen, F. Zhang, Y. X. Sun, and X. M. Shen, "A distributed TDMA scheduling algorithm for target tracking in ultrasonic sensor networks," *IEEE Transactions on Industrial Electronics*, no. 99, 2012.

[24] H. Chen, W. Lou, and Z. Wang, "A novel secure localization approach in wireless sensor networks," *Eurasip Journal on Wireless Communications and Networking*, vol. 2010, Article ID 981280, 12 pages, 2010.

[25] M. Fayyaz, "Classification of object tracking techniques in wireless sensor networks," *Wireless Sensor Network*, vol. 3, pp. 121–124, 2011.

[26] O. Demigha, W.-K. Hidouci, and T. Ahmed, "On energy efficiency in collaborative target tracking in wireless sensor network: a review," *IEEE Communications Surveys & Tutorials*, vol. 99, pp. 1–13, 2012.

[27] O. Younis and S. Fahmy, "HEED: a hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks," *IEEE Transactions on Mobile Computing*, vol. 3, no. 4, pp. 366–379, 2004.

[28] J. Y. Yu and P. H. J. Chong, "A survey of clustering schemes for mobile ad hoc networks," *IEEE Communications Surveys & Tutorials*, vol. 7, no. 1, pp. 32–48, 2005.

[29] H. C. Lin and Y. H. Chu, "Clustering technique for large multihop mobile wireless networks," in *Proceedings of the 51st Vehicular Technology Conference (VTC '00)*, pp. 1545–1549, May 2000.

[30] A. Youssef, M. Younis, M. Youssef, and A. Agrawala, "Distributed formation of overlapping multi-hop clusters in wireless sensor networks," in *Proceedings of the IEEE Global Telecommunications Conference, Exhibition & Industry Forum (GLOBECOM '06)*, December 2006.