Research Article

Energy-Aware Distributed Clustering Algorithm for Improving Network Performance in WSNs

Joon-Ik Kong, 1 Jin-Woo Kim, 2 and Doo-Seop Eom 1

1 Department of Electric Engineering, Korea University, Seoul 136-075, Republic of Korea
2 Research Institute of Information Science and Engineering, Mokpo National University, Mokpo 534-729, Republic of Korea

Correspondence should be addressed to Doo-Seop Eom; eomds@korea.ac.kr

Received 7 November 2013; Revised 26 January 2014; Accepted 30 January 2014; Published 10 March 2014

1. Introduction

Wireless sensor networks (WSNs) have been of interest in a wide range of applications, for example, surveillance, environmental monitoring, and emergency medical response. In general, WSNs are usually composed of a large number of sensor nodes with the same performance intended to gather sensory information and communicate with each node. The sensor nodes are equipped with small, cheap electronic components such as a microprocessor, a wireless transceiver, and the like. However, we need to solve resource constraints of the sensor nodes owing to limited computational capabilities, limited storage, short communication range, and limited power. Most of all, it is important to consider the limited power of a sensor node, because it is typically installed in an inaccessible area or is hard to replace. In addition, battery depletion of a sensor node can have a substantial impact on the lifetime of an entire network.

Energy consumption of a sensor node is classified into three components [1]: (1) the energy consumed by sensing the information the user wants, (2) the energy consumed by data processing in the sensor node, and (3) the energy consumed by data communication between nodes. Energy consumption of data communication accounts for the largest proportion, so we need to decrease the sensor nodes’ energy consumption by reducing the amount of data transmission.

Many researchers have focused on developing hierarchical protocols for WSNs to reduce sensor node energy consumption [2–4]. The representative hierarchical algorithm is the clustering algorithm consisting of cluster heads and cluster members. Cluster heads manage their own cluster members to avoid collisions. In addition, they collect sensed data from cluster members in their region and aggregate them in order to decrease data transmission energy consumption.

Many clustering algorithms have been proposed to increase energy efficiency. However, those algorithms have drawbacks in some applications, such as in detecting unexpected events. These applications have characteristic features as follows: (1) the events rarely occur, (2) explosive sensed data occurs on the nodes detecting the events, (3) the events occur locally, and (4) events move randomly in the sensor field. To increase energy efficiency in such applications,
optimal clusters must be created depending on the event type. For example, Figure 1 shows that optimal clusters using minimal nodes are created when the target moves randomly. It is important that the clusters are not configured before detecting the event to reduce unnecessary energy consumption.

In this paper, we propose an energy-efficient clustering algorithm for detecting unexpected events in WSNs. In particular, we consider that the arbitrary target moves freely in the sensor field. In addition, the proposed algorithm creates a cluster-head election window (CHEW), so the sensor nodes create optimal clusters dynamically using a regional competition scheme. This method has the advantage of being able to create clusters, to ensure flexibility of clusters depending on the movement of the target, and to increase the energy efficiency of the entire network.

The remainder of the paper is organized as follows. Section 2 discusses related work. Section 3 describes our proposed energy-efficient clustering algorithm considering unexpected events in detail. We evaluate the performance of the proposed algorithm as compared with other algorithms through computer simulation in Section 4. Finally, the last section summarizes our conclusions.

2. Related Works

Many researchers have studied ways to improve the performance of WSNs, and many clustering algorithms have been introduced to improve energy efficiency [5–7]. Based on this literature, clustering algorithms can be categorized as static and dynamic. Static clustering algorithms, such as LEACH (Low Energy Adaptive Clustering Hierarchy) [8] and HEED (A Hybrid, Energy-Efficient, Distributed clustering) [9], for example, assume that the sensor nodes always have data to send, so the cluster configurations are not changed until the end of their lifetimes. In contrast, dynamic clustering algorithms assume that clusters are created when queries constructed by the user are received or events occurring in the WSN are detected.

LEACH is the best-known clustering algorithm. LEACH periodically repeats a round consisting of two phases, a setup phase and a steady-state phase. In the set-up phase, the clusters are organized as cluster heads and members. The cluster heads are selected randomly in every round to prevent energy consumption by a specific node. In the steady-state phase, data transmission actually occurs according to time division multiple access (TDMA) scheduled by the cluster head, which aggregates data of its own cluster members to reduce the number of transmissions. HEED selects cluster heads by calculating a numerical formula using the degree of closeness with adjacent nodes and residual energy.

However, for several reasons, static clustering algorithms are not suitable for detecting unexpected events. First, they require control overhead for maintaining clusters even though a relevant event rarely occurs. Second, it is inefficient to group nodes in advance, without considering the movement of a target in the sensor field. Third, when an event occurs on boundaries of clusters, energy efficiency is low, because each cluster detecting the target sends the sensed data to the base station separately. This is called a boundary problem [10] in that the clusters do not share the sensed data when the target moves on their boundaries. To solve these problems regarding unexpected events for static clustering algorithms, researchers have been studying dynamic clustering algorithms [11–14]. Dynamic clustering algorithms create clusters efficiently depending on events, rather than according to a fixed cluster configuration.

The authors in [10] proposed a hybrid clustering algorithm combining features of static and dynamic clustering. Their algorithm mainly performs static clustering to collect the information of the target. When the target moves onto the boundaries of clusters, it creates new clusters centered on the target to solve the boundary problem. If the target moves along the boundaries of clusters, this algorithm works in the same way as a dynamic clustering algorithm, creating clusters in response to the movement of the target.

In [15, 16], the authors presented a dynamic clustering algorithm using specific nodes such as the gateway and the elector. In [15], the gateways are installed initially in the sensor field and they detect the amount of residual energy of neighbor nodes. Then, they choose special designated nodes with high residual energy among the nodes as cluster heads. The cluster heads aggregate data and send it to the base station through the gateways. If the gateways malfunction, the network itself does also. In addition, control overhead persists, because the gateway must periodically collect the information of the remaining energy of nodes to select the cluster head. A modified dynamic version of LEACH is presented in [16]. The elector nodes collect the energy information of neighbor nodes and choose a cluster head based on the amount of their energy remaining. In order to prolong network lifetime, the elector nodes periodically select the next elector nodes and the cluster heads. This algorithm improves energy efficiency over that of LEACH in regard to detecting unexpected events; however, control overhead still occurs owing to the collecting of the energy information of neighbor nodes.

In contrast to the above papers, the authors in [17] proposed a clustering algorithm that creates clusters without designated nodes. It is called ARPEES. All nodes detecting the target exchange the message consisting of the node ID, the amount of residual energy, and information about the data sensed with neighbor nodes. After each node receives every message during a specified period, the node selected by the proposed function becomes the cluster head and creates a TDMA schedule for arranging each node. Although this
method creates clusters without using designated nodes, such as the gateway and the elector, control overhead still occurs due to the exchanging of the message with neighbor nodes. In addition, it is difficult to create optimal clusters, because the target can move while messages are being exchanged to configure the cluster.

In this paper, we focus on a dynamic clustering algorithm for detecting the movements of a target and propose an energy-efficient clustering algorithm. Most clustering algorithms must exchange information between neighbor nodes in order to select good cluster heads. In particular, as in the aforementioned literature they must use designated nodes to collect information of sensor nodes, such as residual energy. However, the proposed algorithm can considerably reduce control overhead and requires no designated nodes to collect the information. In simple terms, the cluster heads are selected by competition among the nodes detecting the target. In addition, the clusters can be created quickly. This can reduce both control overhead and energy consumption.

3. The Proposed Algorithm

3.1. The Characteristics of the Proposed Algorithm. A WSN consists of several hundred or thousand sensor nodes. If a few nodes malfunction due to failure of hardware or lack of energy, the network can encounter serious problems. Therefore, the most important factor in prolonging the network’s lifetime is minimizing the nodes’ energy consumption. Many studies have sought ways to maximize the performance of WSNs by studying algorithms such as MAC or routing. Clustering algorithms appear to be better than normal routing algorithms in terms of energy consumption, because they have the feature that the cluster heads manage transmission of their own members to avoid collision and aggregate data sensed by their own members to reduce transmission of duplicated data.

However, static clustering algorithms must be modified in applications that involve unexpected events, because they are inefficient in terms of energy consumption. Such an application usually has the following characteristics. First, an event occurs only rarely. After the sensor nodes are installed, the application needs to maintain sleep mode to save energy, because event occurrence is unpredictable. However, a static clustering algorithm generates control overhead continuously to maintain cluster configuration. Second, the sensed data occurs in the same node. When an event occurs, only a few sensor nodes detect the event and create data to report to the base station. Third, the event occurs locally. The sensor nodes detecting the event are grouped together in the network, but the event itself can move randomly in the sensor field.

Considering these characteristics, we propose an energy-efficient clustering algorithm for detecting unexpected events in WSNs. This algorithm creates optimal clusters along the movement of the target to improve energy efficiency. It repeatedly constructs clusters through competition among the nodes sensing the event to select the cluster head and destroys clusters when the target leaves the cluster’s sensing range. In addition, we propose a method, CHEW, for making clusters effectively. This is described in the next section.

The features of the proposed algorithm are summarized as follows. The algorithm

(i) maintains clustering algorithms’ energy consumption advantages,
(ii) solves the boundary problem, by constructing clusters based on the positions of the events,
(iii) reduces control overhead considerably, by constructing optimal clusters only when an event happens,
(iv) needs no specific nodes to select a cluster head,
(v) can respond actively to unexpected events, because the clusters are configured by CHEW using the residual energy of a node and the distance from a target to a node.

3.2. Cluster Configuration Process. The proposed algorithm is designed with the following characteristics. A WSN consists of many sensor nodes having the same performance. A sensor node becomes a cluster head or cluster member depending on the situation. In addition, the communication range of a sensor node is greater than its sensing range.

When sensor nodes detect an event in the network, the cluster configuration process shown in Figure 2 is executed. Each sensor node detecting the event individually creates a CHEW, the delay time to transmit an advertisement message about cluster head selection (CH_ADV). CHEW is composed of two parts, as shown in (1). First, the integer part determines the maximum size of the window, with a time value determined according to the residual energy at the node and the distance from the node to the target. Ordinarily, the cluster head concentrates on the energy consumption compared with other nodes. A node with a relatively large amount of remaining energy must be chosen as the cluster head. In addition, the node closest to the target must be

![Figure 2: Flowchart to configure clusters using CHEW.](image-url)
selected as the cluster head, because that node will generate the most data to be transmitted to the base station. This method can achieve the effect of reducing the number of data transmissions and thereby increase energy efficiency. Therefore, CHEW sets a small value when the node has a lot of residual energy and is close to the target. Second, the decimal part is a random value between 0 and 1 whose role is to avoid the collision of data transmission when two or more nodes set the same size window:

\[
\text{CHEW} = \text{INT} \left( \frac{D_{\text{target}}}{E_{\text{battery}}} \right) + \text{Random()} ,
\]

where \( 0 < \text{Random()} < 1 \).

After each node quickly sets its CHEW, they start competition to select a cluster head. For example, there are four nodes detecting the target in Figure 3. Each node creates a CHEW using (1), with nodes A, B, C, and D each setting its own CHEW to 2, 3, 4, and 5, respectively. Each node starts its timer separately during CHEW. With the smallest CHEW in this example, node A wins the competition to become the cluster head and broadcasts a CH_ADV message after a specified time. The others drop the goal of being the cluster head and stop their timers. All nodes that receive a CH_ADV message from the cluster head become cluster members, including the nodes defeated in the competition. Remember that the communication range of the sensor node is greater than its sensing range. Including nodes still not detecting the target increases the likelihood that the target will be detected soon.

However, some problems can still be expected, because the CH_ADV message from the cluster head might not be received owing to packet loss or collision. First, some nodes that competed to select the cluster head might not receive the message. Such nodes are called candidate nodes, because they were candidates to be selected as a cluster head. In this case, the node broadcasts its own CH_ADV message independently after its preset CHEW without considering other nodes. As a result, the cluster head learns of the presence of the candidate node and sends a message to change it to a cluster member, so it belongs to the predetermined cluster.

3.3. Management of the Clusters. In the previous section, we studied the process of cluster formation when nodes detect the target. Now, let us discuss the lifetime of a cluster in full detail. When the target moves, the proposed algorithm repeatedly creates optimal clusters and destroys clusters that no longer detect the target. In terms of energy efficiency, this algorithm maintains at least the clusters that are in a position to detect the target. In addition, it is important that this algorithm operates without a designated node during cluster head selection.

Figure 4 illustrates how the next cluster is created along the movement of the target, with "R" as the radius of communication of a node's range, the circle of radius "R" as the current cluster, and "r" as the radius of a node's sensing range. In Figure 4(a), the shading in the circle indicates the area where nodes in the outside of the cluster detect the target, requiring the formation of a new cluster. In Figure 4(b), the presence of the target in the shaded area starts the process of new cluster formation. For example, assume that two nodes detect the target at the same time. According to the proposed algorithm, each node independently creates a CHEW. Then, the nodes compete with each other in order to be a new cluster head. The node that wins the competition becomes the cluster head and creates a new cluster by broadcasting a CH_ADV message. Repeating this process creates optimal clusters along the movement of the target.

When a cluster is created, we need to consider the isolation problem of cluster disconnection [18]. In this algorithm, some nodes of the current cluster inevitably overlap with cluster members of the new cluster, as shown in Figure 5, because the new cluster head detects the target located in the area of the current cluster in advance to transmit information seamlessly to the base station. Thus, the behavior of the overlapped nodes must be clarified, because the overlapped nodes operate as cluster members associated with two clusters. The overlapped nodes give priority to the existing cluster until it is destroyed. Meanwhile, they maintain their link to new cluster.

After the existing cluster is destroyed, or when a failure occurs in the existing cluster, they recall the preset link to the new cluster to prevent information loss and start to transmit information to the new cluster head. Maintaining these links is an advantage in terms of sharing old information on target...
tracking, because either of the two links can be selected as needed.

A cluster’s lifetime ends when it becomes unnecessary and is destroyed to maximize energy efficiency and to reduce maintenance cost. If all nodes in the cluster fail to detect the target during a specified period of time, then the target must have escaped from the sensing range of the current cluster, so the cluster is destroyed by having the cluster head sending its cluster members a termination message.

In brief, Figure 6 summarizes the lifetime of a cluster. In Figure 6(a), assuming that three nodes detect the target, each node immediately creates its CHEW to compete regionally with the others. Then, one node wins the competition, becomes the cluster head, and creates a cluster for the target. If the target moves, as in Figures 6(b) and 6(c), new clusters will be required to store information continuously. At that time, the regional competition begins among the external nodes of the current cluster regardless of the number detecting the target. In Figure 6(b), the cluster is created easily, because only one node detects the target. In Figure 6(c), two nodes detecting the target compete to select the new cluster head. In addition, old clusters not detecting the target anymore are destroyed. Finally, in Figure 6(d), we can see traces of the clusters along the movement of the target. Thus, optimal clusters can be created rapidly along the movement of the target, using CHEW, and energy efficiency can be increased by destroying unnecessary clusters.

3.4. The Rule of State Transition. All nodes in the proposed algorithm are initially in sleep mode in order to save energy. When they detect events, they wake from sleep mode ready to collect information about the events while ensuring that data transmission is seamless and continuously connected. In this section, we define the rule of state transition, whose purpose is to create new clusters effectively while increasing energy efficiency.

All nodes in the proposed clustering algorithm have three states, active, ready, and sleep mode. Figure 7(a) illustrates the states of nodes around one cluster, drawn in bold. First, active mode is set when the cluster is operating on the nodes. The nodes in this state actually work to collect information. Second, ready mode is set to prepare for the next cluster, in anticipation of event detection. Thus, nodes change their state from sleep mode to ready mode as a result of overhearing messages from the current cluster. Finally, sleep mode is set to save the nodes’ energy while they are not detecting events. In addition, nodes are normally kept in sleep mode, because they cannot predict whether events will be detected or not. Figure 7(b) illustrates state transition on a case-by-case basis;

(i) Ready → Active: change when new clusters are created after sensor nodes detect events,
(ii) Ready → Sleep: change when any nodes belonging to a cluster do not detect events during a specified period,
(iii) Active → Sleep: change when the cluster is destroyed,
(iv) Sleep → Ready: change when nodes outside the cluster overhear the messages of nodes in the cluster.

3.5. Multiple Targets. So far, we have considered the clustering algorithm when one target randomly moves in the sensor field. However, it is possible that multiple targets sometimes exist in the sensor field at the same time. Each target independently moves with different features such as velocity, direction, and the like.

First of all, the proposed algorithm creates clusters immediately after sensor nodes detect targets. They have no interest whether targets exist in other areas or not.
Whenever the sensor nodes detect arbitrary targets, they unconditionally make CHEW and compete to create optimal cluster around the target. Even if multiple targets appear in the sensor field, the operation of the proposed algorithm is the same regardless of the number of them. In particular, when multiple targets freely move in sensor field, there are two kinds of considerations. First, if different targets move into same area, different clusters will be merged as shown in Figure 8(a). For example, we assume that two targets in different locations meet at the same place. Clusters are created on respective targets until they meet. If they locate in same area, they will be in detection range of one cluster. Therefore, one cluster is sufficient to collect data of targets. Second, the cluster is split along movement of each target as shown in Figure 8(b). If two targets in same place move in different directions, two clusters will be created along each target. At this moment, it is required to identify each target in order to maintain unique information of the targets, but this topic
to compare performance objectively. Finally, ARPEES is an adaptive routing protocol using a distributed method and dynamic approach, making decisions without any centralized control.

4.2. Simulation Results and Analysis. Figure 9 presents the traces of created clusters while the target moves in a sensor field deployed on a grid. Black circles represent cluster heads, and sensor nodes in dotted circles represent cluster members associated with each cluster head. Through this process, we can verify that the proposed algorithm creates optimal clusters along the movement of the target. Also, it should be noted that the traces of the created clusters are different on every experiment, because the target freely moves in sensor field.

To create optimal clusters for moving targets, the best cluster heads should be selected before everything else. The CHEW makes it possible to select optimal cluster head among the nodes detecting the target. There are three advantages of clusters using CHEW. First, the cluster head is selected by the method of regional competition without exchanging the messages among nodes. And it can aggregate the sensed data around the target. This reduces the energy consumption caused by duplicated transmission considerably. Second, the cluster head is much closer to the target’s path, because it is more likely to generate a relatively large amount of data than others. Consequently, the network lifetime can be extended by reducing the energy consumed for data transmission. Finally, the cluster configuration time must be faster than the target’s speed. The proposed algorithm can create clusters rapidly using CHEW, because delay caused by message exchange is considerably reduced.

Furthermore, the proposed algorithm can increase energy efficiency as a result of operating minimal clusters. Look at the state of the cluster operation, when the target moves as shown in Figure 10. Dotted circles represent the sensing range of each cluster head. When the target is located at A, cluster 1 is only created. When the target moves at position B, the nodes outside of cluster 1 can detect the target. After they make their own CHEW and compete with each other to select optimal cluster head, cluster 2 is created quickly. This is an important process to ensure the continuity of data created by cluster 1. Now, the two clusters
have been created. Next, when the target moves to position C, cluster 2 continues to collect sensed data, but cluster 1, which can no longer detect the target, is destroyed to prevent unnecessary energy consumption. Similarly, cluster 3 is created at positions D and cluster 2 is destroyed at position E, respectively. Therefore, the proposed algorithm can operate up to two clusters, when one target moves.

In the same way, we experiment by repeatedly expanding the sensing area, as $15 \times 15$, $20 \times 20$, and $25 \times 25$. The speed of the target is set to 10 m/s, the sprint speed of a person. Figure 11 shows the final number of clusters, after the experiments have been repeated several times. The bar graph indicates the number of clusters, which is 5, 7, 9, and 11, respectively, when the target diagonally moves as shown in Figure 9(a). And error bars represent the number of clusters when the target moves randomly. Through the results, we can confirm that the total number of created clusters is varied, because the proposed algorithm offers minimal and optimal clusters depending on the movement of the target and network size. On the other hand, during the operation, the maximum number of clusters concurrently operated remains two for the targets in the same area, as described earlier. That is, the proposed algorithm can increase energy efficiency because it maintains fewer clusters than other algorithms.

Figure 12 presents the ratio of received packets to sent packets while the target moves at various speeds. This simulation intends to confirm how many duplicated packets are received on the destination nodes. In other words, it estimates the performance of data aggregation, the important function of a clustering algorithm, to increase energy efficiency of whole network. Above all, nonclustering algorithm is higher in this regard than other algorithms, because it separately transmits packets that are sensed data about the target without data aggregation. This results in more energy consumption. LEACH has the boundary problem,
because this algorithm does not consider moving target. So, it generates duplicated packets from several clusters, even though each cluster is aggregated data about the target. In addition, there are subtle differences between the proposed algorithm and ARPEES, whether considering the target’s speed or not. They can reduce the duplicated packets due to solving the boundary problem. In particular, the duplicated packets of the proposed algorithm are the least, because it always maintains minimal clusters considering movement of the target.

Figure 13 illustrates the average consumed energy for each algorithm while the target moves at various speeds. In particular, it should be noted that sensor nodes do not know when the target will appear in the sensor field. To reduce unnecessary energy consumption, efficient algorithm is required for the unexpected targets.

In the outcome of this simulation, nonclustering algorithm consumes the highest energy, because it produces a lot of duplicated packets as shown in Figure 12. LEACH consumes a relatively large amount of energy among clustering algorithms. This algorithm periodically requires considerable packets to maintain clusters whether the target exists or not. In addition, this algorithm is apt to unnecessarily operate many clusters to detect the target, because it does not solve the boundary problem. By the way, two algorithms, the proposed algorithm and ARPEES, can increase energy efficiency as a result of solving the boundary problem. Also, they only operate when the target appeared. The proposed algorithm is slightly more efficient than ARPEES. The reasons are that it maintains minimal clusters depending on the movement of the target and it reduces control overhead to create cluster.

Figure 14 shows end-to-end delay for detecting nodes to transmit packets to a sink node. End-to-end delay of the nonclustering algorithm is the least, because the sensor nodes detecting the target immediately transmit data to a sink node. Next, LEACH can quickly transmit data to sink node among clustering algorithm, because it already maintains cluster configuration. It spends the time to aggregate data for the most part. Meanwhile, the proposed algorithm and ARPEES do not have any clusters before the target is detected. After they detect the target, they will create clusters for the target. And the clusters will aggregate data generated by their own cluster members. Therefore, these require a relatively large number of times for data transmission. The difference between the two algorithms is the time needed to configure the cluster. The proposed algorithm can transmit data slightly faster than ARPEES, because it configures clusters through the regional competition without exchanging messages.

5. Conclusion

Previous clustering algorithms have problems with applications involving the detection of unexpected events.
In this paper, we defined the features of unexpected events and proposed an energy-efficient clustering algorithm that can respond quickly to such events with increased energy efficiency. In particular, when an arbitrary target moves, our algorithm creates and destroys clusters, optimally. Because each sensor node detects events individually, and clusters are created through regional competition, this method can dramatically reduce control overhead for maintaining the status information of neighbor nodes, thereby solving the boundary problem. In addition, we verified through simulations that the clusters are optimized along the target’s movement with increased energy efficiency.

**Conflict of Interests**

The authors declare that they have no conflict of interests regarding the publication of this paper.

**Acknowledgment**

This research was supported by the ADD (Agency for Defense Development), Korea.

**References**

[1] H. Y. Shwe, X. H. Jiang, and S. Horiguchi, “Energy saving in wireless sensor networks,” Journal of Communication and Computer, vol. 6, no. 5, 2009.

[2] S. K. Singh, M. P. Singh, and D. K. Singh, “A survey of energy-efficient hierarchical cluster-based routing in wireless sensor networks,” International Journal of Advanced Networking and Applications, vol. 2, no. 2, pp. 570–580, 2010.

[3] S. K. Singh, M. P. Singh, and D. K. Singh, “Routing protocols in wireless sensor networks—a survey,” International Journal of Computer Science and Engineering Survey, vol. 1, no. 2, 2010.

[4] J. N. Al-Karak and A. E. Kamal, “Routing techniques in wireless sensor networks: a survey,” IEEE Wireless Communications, vol. 11, no. 6, pp. 6–28, 2004.

[5] X. Liu, “A survey on clustering routing protocols in wireless sensor networks,” Sensors, vol. 12, no. 8, pp. 11113–11153, 2012.

[6] A. A. Abbasi and M. Younis, “A survey on clustering algorithms for wireless sensor networks,” Computer Communications, vol. 30, no. 14-15, pp. 2826–2841, 2007.

[7] N. K. Mishra, V. Jain, and S. Sahu, “Survey on recent clustering algorithms in wireless sensor networks,” International Journal of Scientific and Research Publications, vol. 3, no. 4, 2013.

[8] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, “Energy-efficient communication protocol for wireless microsensor networks,” in Proceedings of the 33rd Annual Hawaii International Conference on System Sciences (HICSS ’00), pp. 10–19, January 2000.

[9] 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.

[10] Z. Wang, W. Lou, Z. Wang, J. Ma, and H. Chen, “A hybrid cluster-based target tracking protocol for wireless sensor networks,” International Journal of Distributed Sensor Networks, vol. 2013, Article ID 494863, 16 pages, 2013.
Hindawi
Submit your manuscripts at
http://www.hindawi.com