



CRPD: a novel clustering routing protocol for dynamic wireless sensor networks

Shaoqing Wang¹ · Jiguo Yu¹ · Mohammed Atiquzzaman² · Honglong Chen³ · Lina Ni⁴

Received: 09 September 2017 / Accepted: 10 December 2017 / Published online: 12 February 2018
© Springer-Verlag London Ltd., part of Springer Nature 2018

Abstract

A wireless sensor network (WSN) consists of a large number of static or mobile, low-cost, and low-power sensor nodes. And energy is one of the most important factors that should be considered. In this paper, we propose clustering-based routing protocol for dynamic networks (CRPD) to reduce energy consumption and improve energy efficiency through clustering and routing algorithms. The basic idea is to periodically update the network topology and select the node with larger degree and high residual energy as the cluster head to be responsible for data aggregation and transmission. With the nodes moving, joining, and choosing the optimal clustering radius, the energy load of the whole network can be evenly distributed to each sensor node, which can significantly prolong the network lifetime. Extensive simulations show that CRPD is more energy-efficient than the existing protocols.

Keywords Wireless sensor networks · Dynamic · Clustering · Routing protocol · Energy efficiency

1 Introduction

A wireless sensor network (WSN) is a wireless network composed of a group of sensors in an ad hoc mode. It aims to perceive, collect, and process the information of the sensing objects in the geographic area covered by the

network, and transmits the sensed data to one or more sink nodes. For the last two decades, wireless sensor networks have been applied to many fields, such as habitat monitoring [1], target tracking in battlefields [2], structural monitoring [3], and gas monitoring [4]. Specially, with the development of the Internet of Things (IoT), WSNs obtained a sustainable development.

However, the sensor nodes usually only rely on battery power, and the battery cannot be replaced once deployed. In addition to the precious and scarce power resource, there are also some limited resources, such as the processing power, the wireless bandwidth, and the storage space. And the limited resources have posed great challenges to WSN technologies. Therefore, how to reduce energy consumption and get efficient usage of the storage space in the process of collecting and gathering data is particularly important. Furthermore, the neighbor nodes may have strong correlation among the sensed data due to their spatial position, which makes the highly redundant aggregated data occupy an extra part of the storage space and consume more energy. Hence, in order to improve the performance of WSNs, various network technologies (such as compression technology [5] and data aggregation technology [6, 7]) have emerged in recent years.

Essentially, a WSN is a dynamic network. The topology of a WSN may change due to some of the following factors:

✉ Jiguo Yu
jiguoyu@sina.com

Shaoqing Wang
shaoqing_wang@163.com

Mohammed Atiquzzaman
atiq@ou.edu

Honglong Chen
chenhl@upc.edu.cn

Lina Ni
nln2004@163.com

¹ School of Information Science and Engineering, Qufu Normal University, Rizhao, 276826, Shandong, China

² School of Computer Science, University of Oklahoma, Norman, OK 73019-6151, USA

³ College of Information and Control Engineering, China University of Petroleum, Qingdao, 266500, China

⁴ College of Computer Science and Engineering, Shandong University of Science and Technology, Qingdao, 266590, China

- A sensor node may exit from a running network due to its battery power exhaustion or other failures;
- We sometimes need to add new nodes to the network due to actual requirement;
- The sensor nodes, sink nodes, and sensing objects themselves may move in some cases [8–11];
- The changes of environmental conditions may lead to changes of wireless communication link bandwidth, or even temporary interruption.

Therefore, the WSN technology is required to be able to adapt to these dynamic changes, that is, the network should have the function to dynamically update topology. Luckily, clustering, an effective topology control method, has been widely studied. However, there are some differences between the existing dynamic clustering and the clustering in dynamic wireless sensor networks:

- Dynamic clustering [12, 13]:
 - It is assumed that the nodes are static after deployment and no longer move;
 - It does not consider the situation in which the new nodes can join in;
 - There is no case of node death, and the robustness is poor.
- The clustering in dynamic wireless sensor networks:
 - Nodes can move after deployment;
 - It allows new nodes to join into the network;
 - It can deal with the situation of node death in order to ensure the normal work of the network.

Since the communication distance of the node in the network is limited, if a node wants to communicate with the node outside its radio frequency coverage range, it needs intermediate nodes to route. That is to say in a multi-hop environment, a sensor node sends data to the sink through the intermediate sensor nodes. In energy-constrained sensor networks, networks often need energy-efficient routing protocols to transmit data to ensure the data reliability. Reliable routing significantly reduces re-transmission of data, which can reduce energy consumption. Therefore, the sensor nodes need appropriate energy conservation and reliable routing for data transmission.

In this paper, a clustering-based routing protocol for dynamic networks (CRPD) is proposed to solve the above requirements. In WSNs, using clustering technology can help reduce the network traffic and energy consumption, thus extending the lifetime of the entire network. The routing algorithm we adopted selects the nearest node to the target node as the next hop in real time, which not only ensures the selection of the shortest path but also saves the energy of some nodes. In addition, we adopt the confirm

packets to achieve the reliable delivery of data, and thereby improve the reliability of the network.

The rest of the paper is organized as follows: the related work is discussed in Section 2. The working principle and algorithm of the proposed protocol is presented in Section 3. In the Section 4, simulation parameters, results, and analysis are discussed. Finally, Section 5 gives the conclusion and future work.

2 Related work

Many clustering algorithms have been proposed in past few years. In this section, we briefly review the existing clustering algorithms from different perspectives.

- According to the different implementation ways, the clustering algorithms can be divided into centralized and distributed. A centralized clustering algorithm requires the global information of the network and can select a certain number of cluster head nodes with better distribution, but this approach is limited in large-scale networks since the global information of the network cannot be obtained. However, a distributed clustering algorithm does not need to acquire the global information of the network. Instead, the nodes in the network perform the clustering task independently according to the local information, which leads to lower energy consumption and is more suitable for large-scale networks.

LEACH, proposed by Heinzelman et al. in 2000, is the most typical round-robin distributed clustering protocol [14, 15]. In LEACH, each round consists of two phases: cluster establishment and data transmission. In the cluster establishment phase, each node selects a random number between 0 and 1, and if the number is less than a certain threshold, then the node becomes a cluster head. After that, the cluster head broadcasts a message to all its neighbors to inform that it has been a cluster head. After receiving the message, each node decides which cluster to join according to the strength of the received signal, and replies to the cluster head. In the data transmission phase, all nodes in the cluster send data to the cluster head according to TDMA (Time Division Multiple Access) mechanism. The cluster head fuses all the received data and then sends the results to the base station (BS). After a period of continuous operation, the network re-enters the start up phase, the next round of cluster head selection, and re-establishment of clusters, to make the role of the cluster head in the whole network being periodically rotated. The protocol is simple and it does not require large communication overhead. However, the cluster head location is not

uniform and the cluster head selection does not consider energy.

The algorithms proposed in [16, 17] are centralized version of LEACH. They improve LEACH by using central control, that is, the BS is responsible for collecting information (e.g., node energy and location information) from all sensor nodes and selecting the best cluster head. The drawback of this type of protocol is that the clustering process can be very complex and can incur additional overhead. Therefore, the centralized clustering algorithm has weak scalability, and only applies to small- or medium-sized network. Hence, most efficient clustering algorithms (e.g., [18–24]) are distributed.

- According to the layer number of clusters, clustering algorithms can be divided into single-layer clustering and multi-layer clustering. The single-layer clustering divides the network into two layers, in which the cluster heads are as high level and the cluster members are as low level. The whole network is composed of high-level cluster heads and low-level cluster members for each cluster. In a multi-layer clustering, the low-level cluster head usually serves as a member of the high-level cluster head, so it can further reduce the energy consumption, but its implementation is complex and its overhead is often larger.

In [25], a top-down approach is used to construct a multi-layer cluster topology. In the cluster establishment stage, the nodes become the first-layer cluster heads with the probability $p_1(u)$, and the other nodes become the first-layer cluster members. Then, the first layer cluster heads inform its cluster members to carry out the second-level cluster head election, the nodes in the first-layer cluster members could become the second-layer cluster heads with the probability $p_2(u)$, the remaining nodes become the second-layer cluster members, and the same process is done until the number of nodes within a cluster is not more than three. In the data transmission phase, the T-layer cluster members send data to the T-layer cluster head, T-layer cluster heads send the fusion data to the $(T - 1)$ -layer cluster heads and so on, until the one-layer cluster heads send the fusion data to the BS.

In [26], Alkalawi et al. also proposed a cross-layer clustering-based multipath routing protocol. The sink initiates the cluster forming phase through broadcasting a control packet, and then, the node becomes the cluster head according to the strength of the received signal and its power. Cluster heads are divided into different layers; they send data through the upper cluster heads. There are two thresholds in this protocol: upper threshold and lower threshold. And the upper threshold is used to determine which node is a cluster head and

which node is a cluster member. The lower threshold is used to establish a link between cluster heads.

- According to the different application range of clustering algorithm, it can be divided into static and dynamic. Most of the clustering algorithms assume that the nodes in the network are static. Once the network topology is constructed, it will not change. Therefore, this kind of algorithm is only suitable for static networks. However, in some cases of wireless sensor networks, nodes have mobility, such as target tracking [27, 28], and then, the clustering algorithm needs to meet the requirement that the network topology can be updated in real time.

The CMRP protocol, proposed by Sharma et al. [29], is a clustering routing protocol for static networks. This protocol reduces the energy of the sensor nodes by giving sink more responsibility. The sink node is required to collect neighbor information from the sensor nodes and create a neighbor adjacency matrix. Then, it can identify the cluster heads, select the appropriate path, and send the path to the selected cluster heads. Obviously, the cluster construction and path selection are both done by the sink, and it is completed only under the premise of that the sink knows the whole network nodes layout. Therefore, it has a great limitation. In [27], Voronoi diagram is used to realize dynamic clustering, so as to achieve the target tracking. Moreover, an adaptive dynamic cluster-based tracking scheme is also proposed in [28] to track the mobile objects.

Different from the most of the previous protocols which assume that the nodes are static, we propose a novel clustering protocol being suitable for dynamic sensor networks. In addition, cluster heads are in heavy responsibility in the previous protocols, and they are not only responsible for inter-cluster communication but also in charge of intra-cluster communication, which causes the energy consumption rate of cluster heads too fast. In our proposed protocol, the responsibility of cluster heads is reduced; we can obtain a better load balancing and a higher energy efficiency.

3 Clustering routing protocol for dynamic wireless sensor networks

In this section, we describe our clustering protocol with theoretical analysis in detail. The CRPD protocol is a cluster-based protocol which is suitable for dynamic sensor networks. It realizes the network data aggregation and data communication by updating the network topology in real time. And the selection of its communication path is determined by each node itself.

Table 1 Description of states of nodes

State	Description
<i>sleep</i>	The node is in sleep state.
<i>idle</i>	The node is in idle state.
<i>chead</i>	The node is a cluster head node.
<i>member</i>	The node is a cluster member node.

3.1 Network model

Assume that the initial sensor network consists of n sensor nodes and a sink node (i.e., BS). The BS has storage, computing ability and unlimited battery power, which is static after the network deployment. In this paper, we make the coordinate of BS is (0, 200). However, other sensor nodes are randomly deployed in the planar region and can move, that is, the network topology is dynamic. Each node is assigned a unique Id . All nodes are homogeneous, that is to say, the computational, communication capabilities and initial energy are identical and predefined. Furthermore, all nodes know their position coordinates (such as GPS positioning system or positioning scheme, e.g., [30, 31]) and their residual energy, and each sensor node knows the location of the sink node coordinates. We use 40% of the initial energy as the energy threshold.

The states of the nodes, the message types, and the variables involved in the protocol are given in Tables 1, 2, and 3, respectively.

3.2 Clustering routing protocol for dynamic network

CRPD includes four phases: neighbor discovery, cluster head selection and cluster formation, data aggregation and route construction, and re-clustering and re-routing, and the duration of each phase is T_1 , T_2 , T_3 , and T_4 , respectively. In this section, we discuss each phase of

Table 2 Description of control messages

Message	Description
<i>detect</i>	Tuple(Id, x, y, Er)
<i>degree</i>	Tuple($Id, degree$)
<i>i_am_thead</i>	Tuple(Id)
<i>you_are_thead</i>	Tuple(Id)
<i>ordinary</i>	Tuple(Id)
<i>Infor_data</i>	Tuple($Id, data$)
<i>Data_forward</i>	Tuple($Id, data$)
<i>dead</i>	Tuple(Id_i, Id_s) or Tuple(Id_i)
<i>joining</i>	Tuple(Id, x, y, Er)
<i>OK</i>	Tuple(Id, x, y, Er, my_ch)
<i>leave</i>	Tuple(Id_i, Id_s) or Tuple(Id_i)

CRPD in detail. Figure 1 shows the diagram of the four stages.

3.2.1 Neighbor discovery

After the sensor nodes are randomly deployed, the neighbor discovery phase is initialized. Within the time period T_1 , each sensor node i broadcasts the *detect* message in a range of R_a with the power P_{low} (The variable can be seen in Table 3), and the message includes the Id_i , the coordinates (x_i, y_i) , and the remaining energy information of the node (see Algorithm 1 for details). After that, the information is recorded and stored at each received node j to form a neighbor information list. That is, after the neighbor discovery phase, each sensor node has information about all of its neighbor nodes.

The basic idea of this algorithm is that each sensor node has two sets: Nbr and Nbr_INFO . The Nbr set only stores the Id of neighbor nodes; however, the Nbr_INFO set is used to store the Id , coordinate, and residual energy information of its neighbors. Apparently, the sensor nodes need to have a certain storage capacity. First, each sensor node broadcasts the *detect* message to send its own Id , coordinates, and residual energy information to all its neighbors. After that, each sensor node will receive the *detect* messages from its neighbors and then store the information of the message accordingly. Therefore, after completing the neighbor discovery phase, each node has information about all its neighbors.

Algorithm 1 Distribute neighbor discovery

```

1 set of int  $Nbr \leftarrow \phi$ 
2 set of tuples  $Nbr\_INFO \leftarrow \{ \langle Id_i, x_i, y_i, Er_i \rangle \}$ 
3 states sleep, idle, currstate
4 message types detect
5  $currstate \leftarrow sleep$ 
6 if timer interrupt then
7      $currstate \leftarrow idle$ 
8     broadcast detect( $Id_i, x_i, y_i, Er_i$ )
9     if receive detect( $Id_j, x_j, y_j, Er_j$ ) then
10         if  $Id_j \notin Nbr$  then
11              $Nbr \leftarrow Nbr \cup \{ Id_j \}$ 
12              $Nbr\_INFO \leftarrow Nbr\_INFO \cup \{ \langle Id_j, x_j, y_j, Er_j \rangle \}$ 

```

Theorem 1 The time complexity of distribute neighbor discovery algorithm is $O(\Delta)$, and the message complexity is $O(m)$, where Δ is the maximum node degree in the network and m is the number of edges (number of links).

Proof Since the distribute neighbor discovery algorithm is distributed, the time complexity of the entire network is equal to the time complexity of a single node. And

Table 3 Description of variable

Variable name	Description
P_{low}	The transmit power of a node
R_a	Clustering radius
R_c	Communication radius
<i>currstate</i>	To record the current state of a node
<i>Nbr</i>	A set that records all neighbors' <i>Ids</i> within R_a range of the current node
<i>Nbr_INFO</i>	A set that records all neighbors' (<i>Id</i> , <i>x</i> , <i>y</i> , <i>Er</i>) information within R_a range of the current node
<i>received</i>	A set that records the received messages
	To ensure that all its neighbors' messages have been received
<i>my_heads</i>	A set which records the nodes to be sent the <i>you_are_head</i> message
<i>Nbr_degrees</i>	A set which records the degree of all neighbors
E_r	The remaining energy of the current node <i>i</i>
$E_{threshold}$	Energy threshold
<i>candidate_heads</i>	The candidate cluster heads set of the current node
$degree_i$	The degree of the current node <i>i</i>
Id_i	The Id of the current node <i>i</i>
<i>my_ch</i>	The cluster head of the current node
<i>Data</i>	To store the data gathered from cluster members
$Distance_i$	To record the distance between the node and its neighbors to the sink
x_{sink}, y_{sink}	The <i>x</i> -coordinate, <i>y</i> -coordinate of sink node
<i>next_hop</i>	To record the next-hop node
E_d	The energy threshold of a dying node
<i>Route_Nbr</i>	A set that records all neighbors' information within R_c range

each node needs to receive *detect* messages from all its neighbors, then the worst case time complexity of a single node is $O(\Delta)$ (Δ is the maximum node degree in the network). Thus, the time complexity of the algorithm is $O(\Delta)$.

Note that each node sends a *detect* message, and there are at most two *detect* messages on one edge. Therefore, the total number of messages is $2m$, that is, $O(m)$. \square

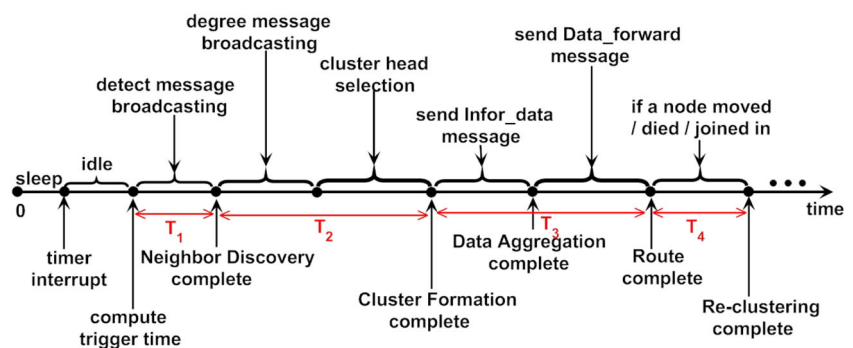
3.2.2 Cluster head selection and cluster formation

After completion of the neighbor discovery phase (T_1), the phase of cluster-head selection and cluster formation is

started, and its duration is T_2 . As assumed above, all sensor nodes have the same energy E_r at first, and each node knows its own residual energy E_r . The energy threshold $E_{threshold}$ is 40% of the initial energy. We use the following principles to select a number of cluster heads:

1. We select the nodes that have the largest degree (i.e., the node that has the largest number of neighbors) compared to all its neighbors.
2. The residual energy E_r is greater than the energy threshold $E_{threshold}$.
3. If the E_r of the node with the largest degree is not greater than $E_{threshold}$, then we select another node with

Fig. 1 The diagram of the dynamic clustering routing protocol



the largest degree among its neighbors, and the residual energy E_r of the selected node needs to be greater than $E_{threshold}$, to be the candidate cluster head.

- Any two cluster heads cannot be neighbors of each other. From the above principles, the selection of cluster head depends on two factors: one is the node degree and the other is the residual energy E_r of the nodes.

After the cluster head selection is completed, the cluster formation phase is started. The concrete implementation process is shown in Algorithm 2. The algorithm consists of two phases: the first stage is to obtain the degree of all the neighbors. The second stage is the cluster formation. First, each node is in the *sleep* state; when the timer interrupts, the node is awakened to the *idle* state. Next, the cluster head is selected according to the cluster head selection principles. If a node has a maximum degree in all its neighbors and its remaining energy is greater than $E_{threshold}$, then we select the node as the cluster head and mark its state as *head* state, and the node broadcasts an *i_am_thead* message. In addition, if the node is a node with the largest degree among all its neighbors but its residual energy is not greater than $E_{threshold}$, then we select its neighbor node (possibly more than one) with the largest degree among its neighbors and the remaining energy of the selected node(s) is/are greater than $E_{threshold}$, to be the candidate cluster head(s). Next, the current node (with the largest degree among all its neighbors but its residual energy is not greater than $E_{threshold}$) sends a *you_are_thead* message to the candidate cluster head node to inform the node that it may be a cluster head, and sends an *ordinary* message to other neighbors. And at the same time, the current node marks its own state as *member* state.

For a node that received a *you_are_thead* message, it needs to further decide whether it has received an *i_am_thead* message; if not, it marks its own state as *thead* state and broadcasts an *i_am_thead* message. Otherwise, it broadcasts an *ordinary* message. Note that, since it is a distributed algorithm, there may be a case that a node and one of its neighbor may received a *you_are_thead* message at the same time and both of them did not receive the *i_am_thead* message. To solve this problem, we introduce the intelligent waiting policy. After receiving the *you_are_thead* message, the two nodes that did not receive the *i_am_thead* message wait for a certain period of time according to the intelligent waiting policy. After the certain period of time, the two nodes determine whether they can become a cluster head according to whether they received an *i_am_thead* message. The waiting time of the intelligent waiting policy is calculated as follows:

$$t_i = \lambda \times \left(1 - \frac{D_i}{R_c}\right) \times \left(1 - \frac{E_r_i}{E}\right), \tag{1}$$

where D_i is the relative distance between the node and the sending node from which the *you_are_thead* message received, R_c is the communication range of the node, and E_r_i is the residual energy of the node. E represents the initial energy of the node, and λ is the time coefficient used to prevent both nodes from simultaneously broadcasting *i_am_thead* messages. The value of λ is determined according to the specific requirements of the waiting.

In addition, other nodes that do not satisfy the cluster-head conditions and did not receive the *you_are_thead* message broadcast the ordinary message. The node received the *i_am_thead* message records the information of the *candidate_thead* set, so that it can select a node with the largest remaining energy from the *candidate_thead* set as its own cluster head, and then becomes a member of the cluster, marking its state as *member*. Note that when there are nodes with the same maximum residual energy, we choose the one whose *Id* is larger. After the algorithm is finished, the sensor nodes in the network are in one of *thead* and *member* states. The FSM of cluster formation is shown in Fig. 2. Figure 3 shows the process of cluster head selection and cluster formation.

Here, we give an example to further illustrate the algorithm, as shown in Fig. 4. We assume that the residual energy of all the nodes satisfies $E_r > E_{threshold}$, except node 7, and the remaining energy of node 5 is greater than node 1. According to the Algorithm 2, we choose the cluster head nodes 1, 5, and 8. The reason for the selection of the node 8 is that the remaining energy of node 7 does not satisfy the principle 2, then the node 7 selects a node with the largest degree and the residual energy greater than $E_{threshold}$ from its neighbors (3, 6, 8) as its cluster head. However, only the node 8 finally becomes the cluster head according to the intelligent waiting policy. As shown in Fig. 4, cluster head nodes 1, 5, and 8 broadcast an *i_am_thead* message to their neighbors. In particular, the node 7 sends a *you_are_thead* message to nodes 3, 6, and 8, and sends an *ordinary*

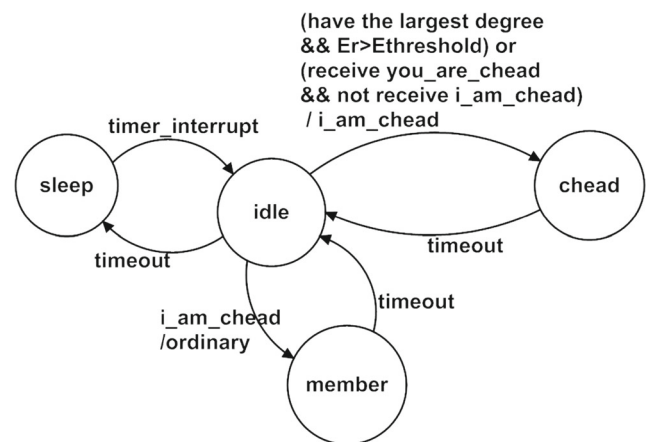


Fig. 2 The FSM of cluster formation

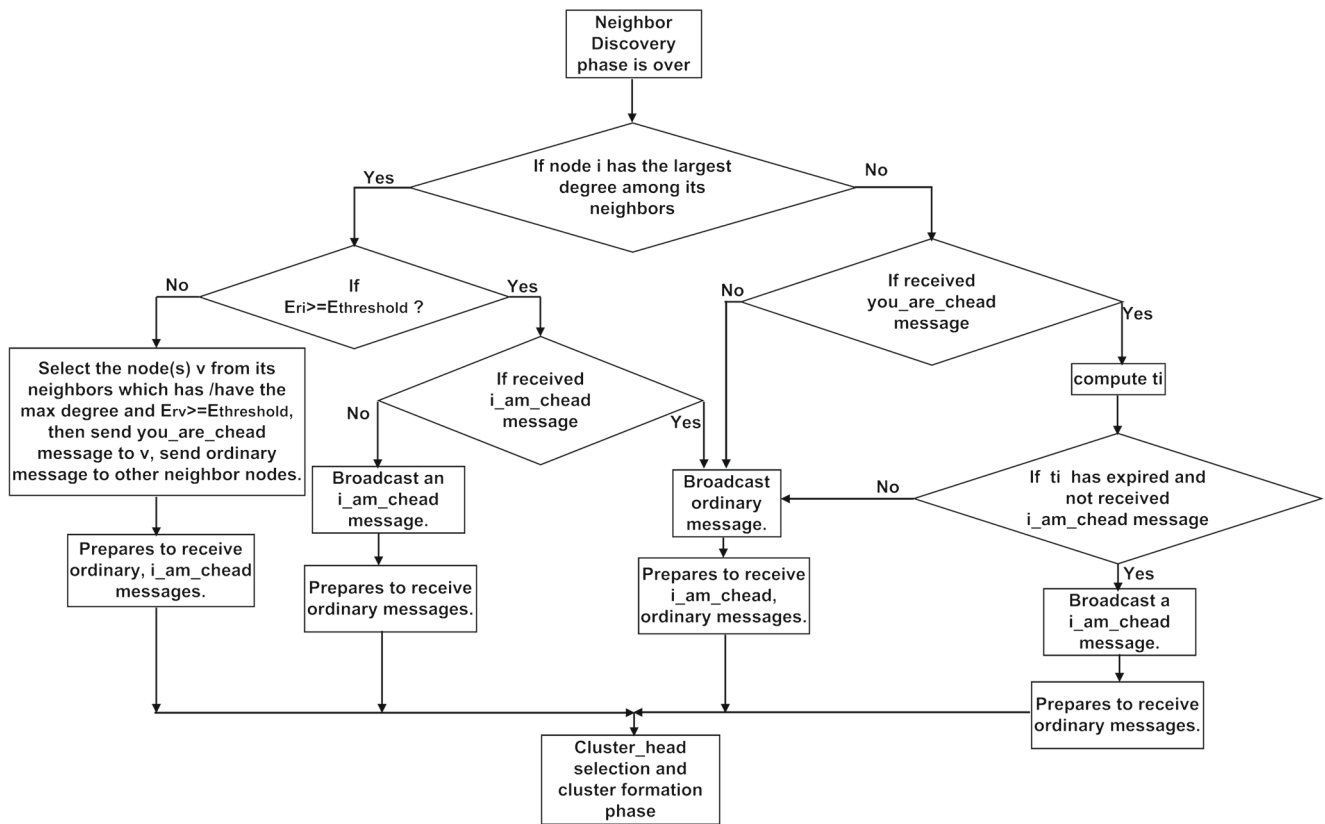


Fig. 3 The process of cluster head selection and cluster formation

message to node 2. Other nodes broadcast an *ordinary* message. For the sake of simplicity, a part of the *ordinary* messages in the figure is not shown. After the algorithm is finished, three clusters (green dashed circle) are formed, as shown in the figure. The *thead* nodes are 1, 5, and 8. The *member* nodes are 2, 3, 4, 6, 7, 9, 10, 11, 12, and 13.

Theorem 2 *Completing the algorithm 2 only needs $O(1)$ rounds. And the message complexity of Algorithm 2 is $O(m)$, where m is the number of edges.*

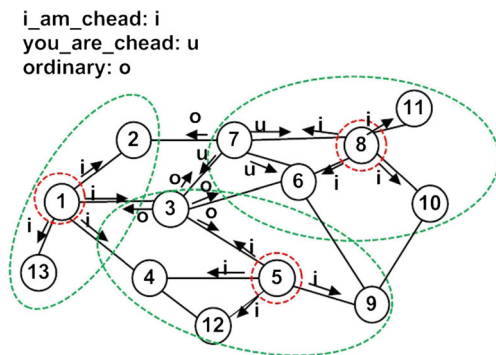


Fig. 4 An example of cluster formation

Proof For each node, the operation of sending, receiving, and processing messages is called a round. The nodes that determine themselves as cluster heads (such as nodes 1 and 5 in Fig. 4) send the *i_am_thead* message and receive *ordinary* messages from their neighbors. The nodes (such as the node 7 in Fig. 4), which have largest degree but do not satisfy the energy principle, will send the *you_are_thead* message to the neighbor nodes which have the largest degree and satisfy the energy principle, and send the *ordinary* message to other nodes, and they will receive the *i_am_thead* and *ordinary* messages from neighbors; in addition, the member nodes send *ordinary* messages, and receive *i_am_thead*, *ordinary* and *you_are_thead* messages. To sum up, each node only needs to send, receive, and process messages only once; after that, it can determine its own state, that is, it only needs $O(1)$ round.

There are at most two *degree* messages on each edge at the stage of getting all the degree of neighbors. At the cluster formation stage, there are also at most two messages per edge, which may be *i_am_thead* and *ordinary*, *you_are_thead* and *ordinary*, *you_are_thead* and *i_am_thead*, *i_am_thead*, and *ordinary* or *ordinary* and *ordinary*. Therefore, there are at most four messages per edge; then, the message complexity of this algorithm is $O(4m)$, that is $O(m)$. □

Algorithm 2 Cluster_head selection and cluster formation

```

1 set of int  $received \leftarrow \phi$ ,  $cheads \leftarrow \phi$ ,  $my\_cheads \leftarrow \phi$ ,
   $Nbr\_degrees \leftarrow \phi$ ,  $Er_i$ ,  $E_{threshold}$ ,  $candidate\_heads \leftarrow \phi$ 
2 int  $degree_i \leftarrow |Nbr|$ ,  $count \leftarrow 0$ ,  $Id_i$ ,  $my\_ch \leftarrow 0$ 
3 states  $chead$ ,  $member$ ,  $currstate$ 
4 message types  $degree$ ,  $i\_am\_chead$ ,  $ordinary$ ,  $you\_are\_chead$ 

```

```

phase 1: Get the degree of all neighbors.
5 broadcast  $degree(Id_i, degree_i)$ 
6 while  $count < degree_i$  do
7   if  $receive\ degree(Id_j, degree_j)$  then
8      $Nbr\_degrees \leftarrow Nbr\_degrees \cup \{degree_j\}$ 
9    $count = count + 1$ 

```

```

phase 2: Cluster formation phase.
10 if  $degree_i = Max\{Nbr\_degrees \cup degree_i\}$  then
11   if  $Er_i \geq E_{threshold}$  then
12     if  $not\ receive\ i\_am\_chead$  then
13       broadcast  $i\_am\_chead$ 
14        $currstate \leftarrow chead$ ;  $my\_ch = Id_i$ 
15   else
16     for  $every\ u \in Nbr$  do
17       if  $Er_u \geq E_{threshold}$  then
18          $cheads = cheads \cup \{u\}$ 
19      $my\_cheads = find\_MaxDegree(cheads)$ 
20     for  $all\ v \in my\_cheads$  do
21       send  $you\_are\_chead$  to  $v$ 
22     send  $ordinary$  to  $Nbr \setminus my\_cheads$ 
23 else
24   if  $receive\ you\_are\_chead$  then
25      $received \leftarrow received \cup \{j\}$ 
26      $t_i \leftarrow \lambda \times (1 - \frac{D_i}{d_0}) \times (1 - \frac{Er_i}{E})$ 
27     if ( $t_i$  has expired &&  $not\ receive\ i\_am\_chead$ ) then
28       broadcast  $i\_am\_chead$ 
29        $currstate \leftarrow chead$ ;  $my\_ch = Id_i$ 
30     else
31       if  $receive\ i\_am\_chead$  then
32          $received \leftarrow received \cup \{j\}$ 
33 else
34   broadcast  $ordinary$ 
35 while  $received \neq Nbr$  do
36   receive  $msg(j)$ 
37   if  $msg(j).type = i\_am\_chead$  then
38      $candidate\_heads \leftarrow candidate\_heads \cup \{j\}$ 
39    $received \leftarrow received \cup \{j\}$ 
40 if  $candidate\_heads \neq \phi$  then
41    $my\_ch = found\_MaxEr(candidate\_heads)$ 
42 if  $Id_i \neq my\_ch$  then
43   if  $Id_i \in Nbr(my\_ch)$  then
44      $currstate \leftarrow member$ 

```

3.2.3 Data aggregation and route construction

After the cluster selection and cluster formation phase (T_2), it is the phase of data aggregation and route construction,

and the duration of this phase is T_3 . Based on the formed cluster, each sensor node sends the sensed data to the cluster head, and then, the cluster head sends the data to the sink. The path selection is done by the distributed Algorithm 3. The sink and sensor nodes all know their position coordinates, and the sink is stationary, while the sensor nodes can move. In addition, each node knows the position of the sink, so we choose a node, closest to the sink, from the neighbors of the node as the next hop until the data is sent to the sink. For the next-hop node selection,

1. Select from the communication range of the current node.
2. Select the node closest to the sink.

Algorithm 3 Data aggregation and route construction

```

1 set of int  $Data \leftarrow \phi$ ,  $Id_i$ ,  $Nbr$ ,  $Route\_Nbr$ ,  $received \leftarrow \phi$ ,
   $Distance_i \leftarrow \phi$ 
2 int  $my\_ch$ 
3 states  $chead$ ,  $member$ ,  $currstate$ 
4 message types  $Infor\_data$ ,  $Data\_forward$ ,  $ack$ ,  $ACK$ 

```

```

5 Calculate the distance to the sink.
6 for  $all\ u \in \{Route\_Nbr \cup Id_i\}$  do
7   compute  $dis = \sqrt{(x_{sink} - x_u)^2 + (y_{sink} - y_u)^2}$ 
8    $Distance_i \leftarrow Distance_i \cup \{< Id_u, dis >\}$ 

```

```

9 The cluster members aggregate the data to the cluster heads.
10 if  $currstate = member$  then
11   send  $Infor\_data(Id_i, data)$  to  $my\_ch$  when each  $time_l$ 
    ends until received  $ack$  message
12 if  $currstate = chead$  then
13   while  $received \neq Nbr$  do
14     if  $receive\ Infor\_data(Id_j, data)$  then
15        $Data \leftarrow Data \cup \{data\}$ 
16        $received \leftarrow received \cup Id_j$ 
17       send  $ack$  to  $Id_j$ 
18   execute the Algorithm 4

```

```

19 Route.
20 while ( $T_3$  has not expired) do
21   if  $currstate = member$  then
22     if  $receive\ Data\_forward(Id_j, data)$  then
23       send  $ACK$  to  $Id_j$ 
24       execute the Algorithm 4
25   if  $currstate = chead$  then
26     if  $receive\ Data\_forward(Id_j, data)$  then
27       send  $ACK$  to  $Id_j$ 
28       execute the Algorithm 4

```

Before executing Algorithm 3, we need to get the neighbor set in the R_c range of each node at first, so we use the idea of Algorithm 1 to send the *detect* message to get the route set $Route_Nbr$. After that, we execute Algorithm 3. The basic idea of the algorithm is that the *member* nodes gather data to their *chead* nodes, and then, the *chead* nodes send the data to the sink according to the route. In the execution of this algorithm, we adopt the *ack* and *ACK*

confirmation messages to ensure the reliable delivery of data. In addition, in order to avoid the same message being sent repeatedly for many times, we set a time value is *time1*, which is used for saving energy.

```

Algorithm 4 Select next_hop node
1 set of int  $Id_i, Route\_Nbr, Distance_i$ 
2 int  $x_{sink}, y_{sink}, next\_hop$ 
3 message types  $Data\_forward, ACK$ 
4 if  $Distance_i \neq \emptyset$  then
5    $next\_hop = find\_MinDistance\_Id(Distance_i)$ 
6   if  $next\_hop \neq Id_i$  then
7     send  $Data\_forward(Id_i, data)$  to  $next\_hop$  when
       each  $time1$  ends until received ACK message
8   else
9     send  $Data\_forward(Id_i, data)$  to sink when each
        $time1$  ends until received ACK message
    
```

Taking Fig. 4 for an example, we show the analysis of Algorithm 3 in Fig. 5. We choose a *thead* node (node 1) to explain. First, the *thead* node 1 receives the *Infor_data* message from all its cluster member nodes and then sends an *ack* message to the neighbors. After that, node 1 sends a *Data_forward* message to node 4, where the node 4 is the nearest node which is close to the sink in its neighbors (including itself). After receiving the *Data_forward* message, node 4 sends an *ACK* to node 1, and then forwards the *Data_forward* message to node 5 which is the closest neighbor to sink. Similarly, node 5 receives the *Data_forward* message, then sends an *ACK* to node 4, and forwards the *Data_forward* message to node 9. After receiving the *Data_forward* message, node 9 sends an *ACK* to node 5. And after finding that the node nearest to the sink among its neighbors is itself, the node 9 sends the *Data_forward* message to the sink directly. Once the sink receiving the *Data_forward* message from the node 9, it will reply an *ACK* message, and then, the route is finished.

Theorem 3 The time complexity of Algorithm 3 is $O(\Delta + \lceil d/R_c \rceil)$ rounds, and the message complexity is $O(m)$,

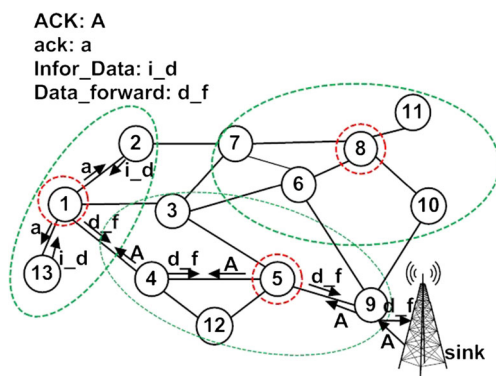


Fig. 5 An example of data aggregation and route construction

where Δ is the maximum degree of the nodes in the network, d is the furthest distance from the sink in all cluster heads, R_c is the communication range of the node, and m is the number of edges.

Proof The Algorithm 3 consists of two phases: data aggregation phase and routing phase. For the data aggregation phase, each cluster member sends an *Infor_data* message to its cluster head and will receive an *ack* message, which only needs one round. The cluster head node receives *Infor_data* messages and sends *ack* messages, which requires Δ rounds. Therefore, completing the data aggregation stage needs $O(\Delta)$ rounds. For the routing phase, each node on the path receives a *Data_forward* message and sends an *ack* message. In other words, how many nodes on the path determines how many rounds we need. The hops of the longest route in the network are $\lceil d/R_c \rceil$. Therefore, completing the routing phase needs $O(\lceil d/R_c \rceil)$ rounds. In conclusion, this algorithm needs $O(\Delta + \lceil d/R_c \rceil)$ rounds.

From the analysis above, in the data aggregation phase, each edge has two messages: *Infor_data* message and *ack* message. In the routing phase, only the links which compose to the route will have two messages: *Data_forward* message and *ACK* message. So there are at most four messages for one edge. Therefore, the message complexity of this algorithm is $O(4m)$, i.e., $O(m)$. \square

3.2.4 Re-clustering and re-routing

The nodes in the network have mobility, which makes the topology of the network change dynamically. In addition, sensor nodes may not work properly due to the energy depletion or other failures; sometimes, new sensor nodes need to be added into the network to meet the work requirements, all of these also can cause the topology of the network to change. So it needs to update the network topology in real time, to ensure the normal network communication. For the network changes, we divided it into three situations:

- (1) *Node death*: Once the residual energy of a node is less than or equal to E_d , then the node firstly needs to judge whether it is a cluster head node. If it is, then this node needs to select another node to replace itself as the cluster head at first. And the selected node is the nearest neighbor node to the current node, and it also satisfies the energy threshold. Secondly, it needs to broadcast *dead*(Id_i, Id_s) messages to its neighbors, where Id_i is the *Id* of the dead node and Id_s is the *Id* of the alternative cluster head node. If the dead node is not a cluster head, then the *dead* message only has the first parameter. The node received the *dead* message firstly determines whether the Id_i is its own cluster

head's Id . If it is, it will replace its cluster head's Id with Id_s , and then removes the Id_i from its neighbor list. Otherwise, it directly deletes Id_i from its neighbor list.

- (2) *Node join*: If a new node is added to the network, then the node broadcasts a *joining*(Id_i, x_i, y_i, Er_i) message to its neighbors. The node that received the message updates its neighbor list and replies an *OK*($Id_i, x_i, y_i, Er_i, my_ch_i$) message, where my_ch_i is the cluster head Id of the node i . After receiving the *OK* message, the newly added node stores the neighbor information and selects the node closest to itself as its cluster head from the received information my_ch_i .
- (3) *Node movement*: If a node moves, we must know whether it is a cluster head node or not at first. If it is an ordinary node, then the node broadcasts a *leave*(Id_i) message to inform the nodes in its original area when it starts to move, after that the node received the *leave*(Id_i) message will delete the sending node from its neighbor list. When a node movement stopped, the node broadcasts the *joining*(Id_i, x_i, y_i, Er_i) message to its neighbors in the new area and then operates according to situation (2). If the moved node is a cluster head, then the node chooses the nearest node whose energy satisfies the energy threshold from its neighbors to replace itself as the cluster head, and then broadcasts the *leave*(Id_i, Id_s) message to its original area. The node received the *leave*(Id_i, Id_s) message will determine whether Id_i is its own cluster head; if it is, it will update its own cluster head to Id_s and then removes Id_i from its neighbor list. If it is not, the Id_i will be directly removed from the current node's neighbor list. When the movement of the node is stopped, a *joining*(Id_i, x_i, y_i, Er_i) message is broadcasted to the new area, and then the algorithm operates according to situation (2).

Theorem 4 *The time complexity of re-clustering algorithm is $O(n)$ rounds and the message complexity is $O(m)$, where n is the number of current network nodes and m is the number of edges.*

Proof When “node death” occurs, the dead node broadcasts the *dead* message, and its neighbors receive *dead* messages. In this process, nodes either send messages or receive messages, which only needs one round to complete.

When “node join” occurs, the new joining node broadcasts the *joining* message and receives the *OK* message, and its neighbor nodes receive the *joining* message and sends an *OK* message. Completing this process only needs one round.

Algorithm 5 Re-clustering

```

1 Situation (1):
2 if  $Er_i \leq E_d$  then
3   if  $Id_i$  is a cluster head then
4     broadcast dead( $Id_i, Id_s$ )
5   else
6     broadcast dead( $Id_i$ )
7 if received dead( $Id_j, Id_s$ ) or dead( $Id_j$ ) then
8   if  $my\_ch = Id_j$  then
9      $my\_ch = Id_s$ 
10   $Nbr \leftarrow Nbr \setminus Id_j$ 

```

```

11 Situation (2):
12 if a new node  $Id_i$  come then
13   broadcast joining( $Id_i, x_i, y_i, Er_i$ )
14 if received joining( $Id_j, x_j, y_j, Er_j$ ) then
15    $Nbr \leftarrow Nbr \cup \{Id_j\}$ 
16    $Nbr\_INFO \leftarrow Nbr\_INFO \cup \{<Id_j, x_j, y_j, Er_j>\}$ 
17   send OK( $Id_i, x_i, y_i, Er_i, my\_ch_i$ ) to  $Id_j$ 
18 if received OK( $Id_j, x_j, y_j, Er_j, my\_ch_j$ ) then
19    $my\_chs \leftarrow my\_chs \cup \{my\_ch_j\}$ 
20  $my\_ch = select\_ch(my\_chs)$ 

```

```

21 Situation (3):
22 if a cluster head will move then
23   broadcast leave( $Id_i, Id_s$ ) before moving
24   broadcast joining( $Id_i, x_i, y_i, Er_i$ ) after moved
25 if other nodes will move then
26   broadcast leave( $Id_i$ ) before moving
27   broadcast joining( $Id_i, x_i, y_i, Er_i$ ) after moved
28 if received leave( $Id_j, Id_s$ ) or leave( $Id_j$ ) then
29   if  $my\_ch = Id_j$  then
30      $my\_ch = Id_s$ 
31    $Nbr \leftarrow Nbr \setminus Id_j$ 
32 if received joining( $Id_j, x_j, y_j, Er_j$ ) then
33    $Nbr \leftarrow Nbr \cup \{Id_j\}$ 
34    $Nbr\_INFO \leftarrow Nbr\_INFO \cup \{<Id_j, x_j, y_j, Er_j>\}$ 
35   send OK( $Id_i, x_i, y_i, Er_i, my\_ch_i$ ) to  $Id_j$ 
36 if received OK( $Id_j, x_j, y_j, Er_j, my\_ch_j$ ) then
37    $my\_chs \leftarrow my\_chs \cup \{my\_ch_j\}$ 
38  $my\_ch = select\_ch(my\_chs)$ 

```

When “node movement” occurs, the node broadcasts a *leave* message before leaving, and its neighbors receive *leave* messages, so the process only needs one round before the node leaving. After the node enters the new area, it broadcasts the *joining* message and receives the *OK* message, while its new neighbor nodes receive the *joining* message and send the *OK* message, which needs a round to complete. Therefore, for the whole movement of the node, it needs two rounds to be finished.

Based on the above analysis, we now assume that the total number of times for all three cases is a ($0 \leq a \leq n$), so in the worst case (i.e., every time is the node movement), the re-clustering algorithm takes $2a$ rounds, and because of $0 \leq a \leq n$, it needs at most $O(n)$ rounds.

Table 4 Simulation parameter

Parameter name	Value
Network area	200 × 200
Number of sensor nodes	100–500
The coordinate of sink (BS)	(0,200)
Data packet size	512 bytes
Broadcast packet size	20 bytes
E_{elec}	50 nJ/bit
ϵ_{fs}	10 pJ/bit/m ²
ϵ_{mp}	0.0013 pJ/bit/m ⁴
d_0	86 m
E	2 Joules
EDA	5 nJ/bit
α	1/150
β	3/50

From the analysis above, we can get that there is only one message on one edge in the case of “node death.” In the case of “node join,” there are two messages on one edge (joining message and OK message). In the case of “node movement,” there is only one leave message on one edge

before the node moving, while after the movement, there are two messages on one edge, which is similar to the case of “node join.” In summary, we can get that in the worst case there are two messages on one edge, so the message complexity of re-clustering algorithm is $O(2m)$, that is, $O(m)$. □

4 Performance evaluation

4.1 Simulation parameters

We use the MATLAB platform to evaluate the performance of the protocol. We simulate 100–500 nodes randomly deployed in the square region $M \times M$, where $M = 200$. For the same number of nodes, we randomly generate ten network topologies, run the protocol algorithm respectively, and then take the average as the simulation result. For the same network topology, we adopted the average of the first 100 rounds of experimental results as the simulation result. In addition, the BS is located at (0, 200). Specific simulation parameters are shown in Table 4, where E is the initial energy of the node, α is the probability of “node joining,” and β is the probability of “node movement.”

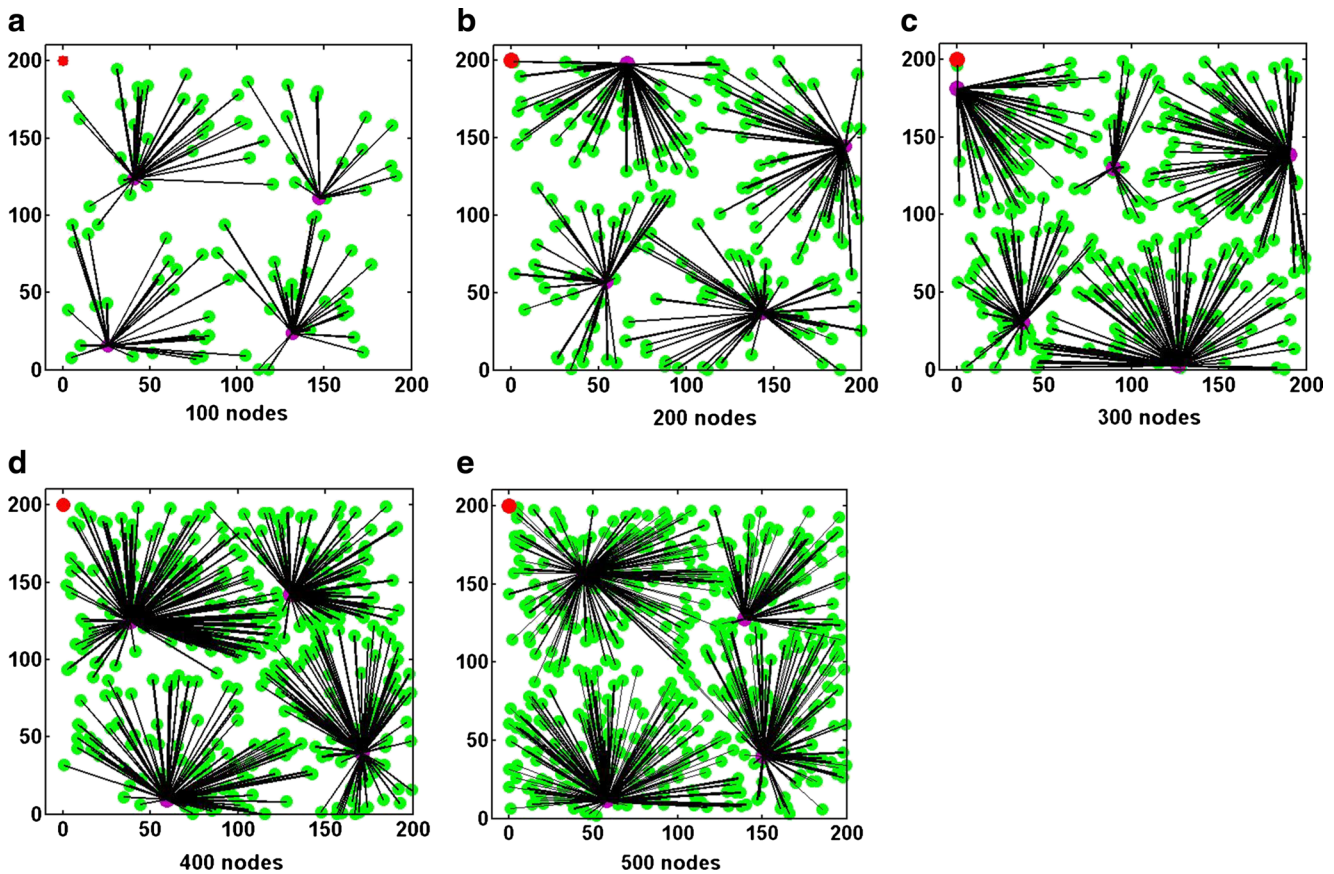


Fig. 6 100–500 node clustering diagram

4.2 Results and analysis

4.2.1 Simulation results

First, we simulate the clustering of 100–500 nodes, as shown in Fig. 6. The (a), (b), (c), (d), and (e) in the figure are the clustering results of 100, 200, 300, 400, and 500 nodes, respectively. And the red dot in the upper left corner is the BS.

After the data aggregation and routing phase, the routing diagram for the five scenarios is shown in Fig. 7. The red arrows are the routing paths.

It can be seen from Fig. 5 that the number of cluster member nodes connected by a cluster head node increases significantly with the increase of the node density in the case of the same size of the network region, which will make the cluster head node consume too much energy during the intra-cluster communication and be dead prematurely. Therefore, in order to reduce the burden of the cluster head nodes in the dense network, we can adjust the clustering radius according to the density of the network and control the cluster size, which can avoid the premature death of the cluster head to prolong the network lifetime.

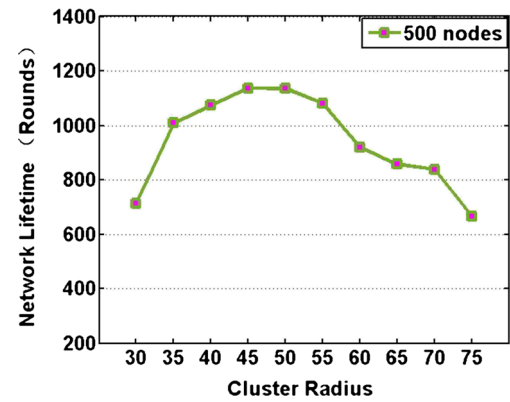


Fig. 8 The variation trend of network lifetime as the cluster radius changed

For a 500-node dense network, we use different values as the cluster radius and plot the variation trend of the network lifetime as the cluster radius change, as shown in Fig. 8. From the figure we can see that when the cluster radius is between 45 and 50, the network lifetime is the longest. Therefore, we can adjust the optimal cluster radius to achieve the purpose of extending the network lifetime.

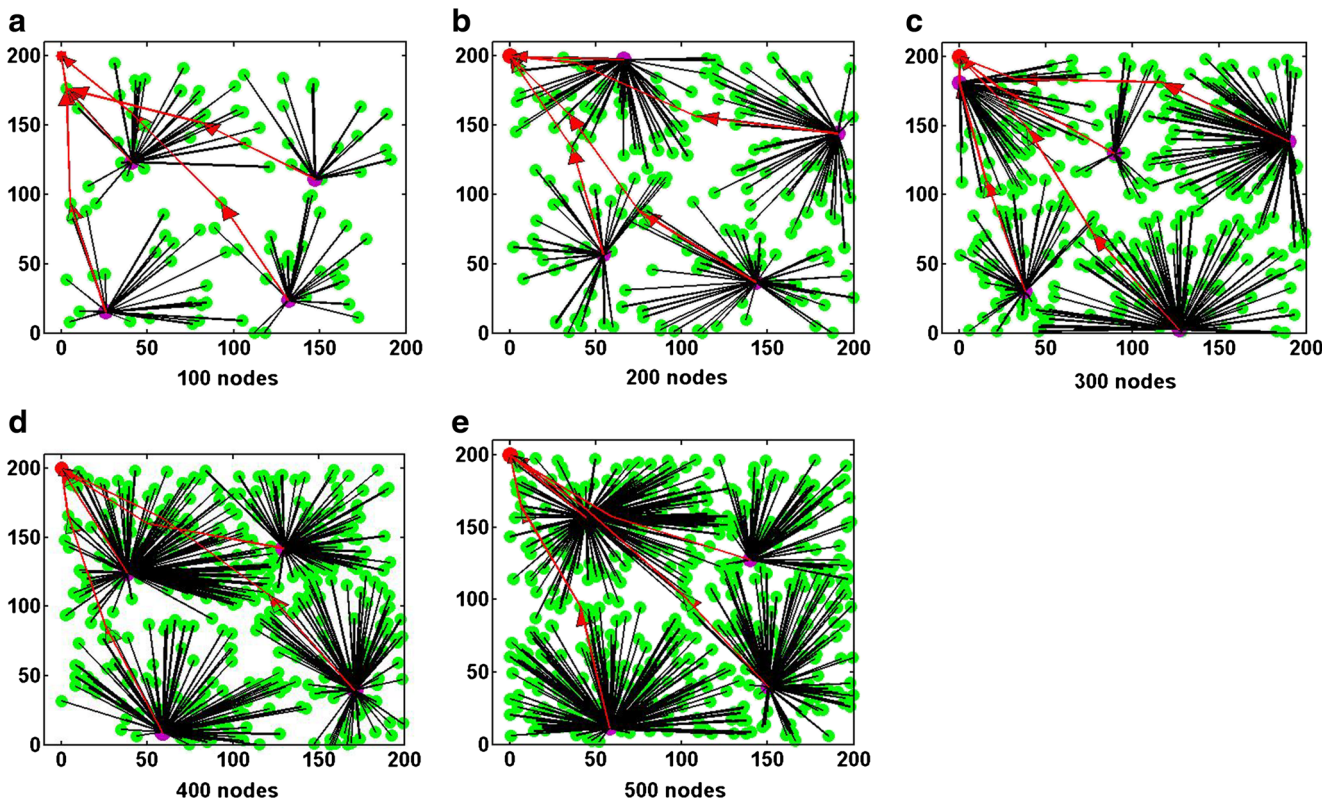
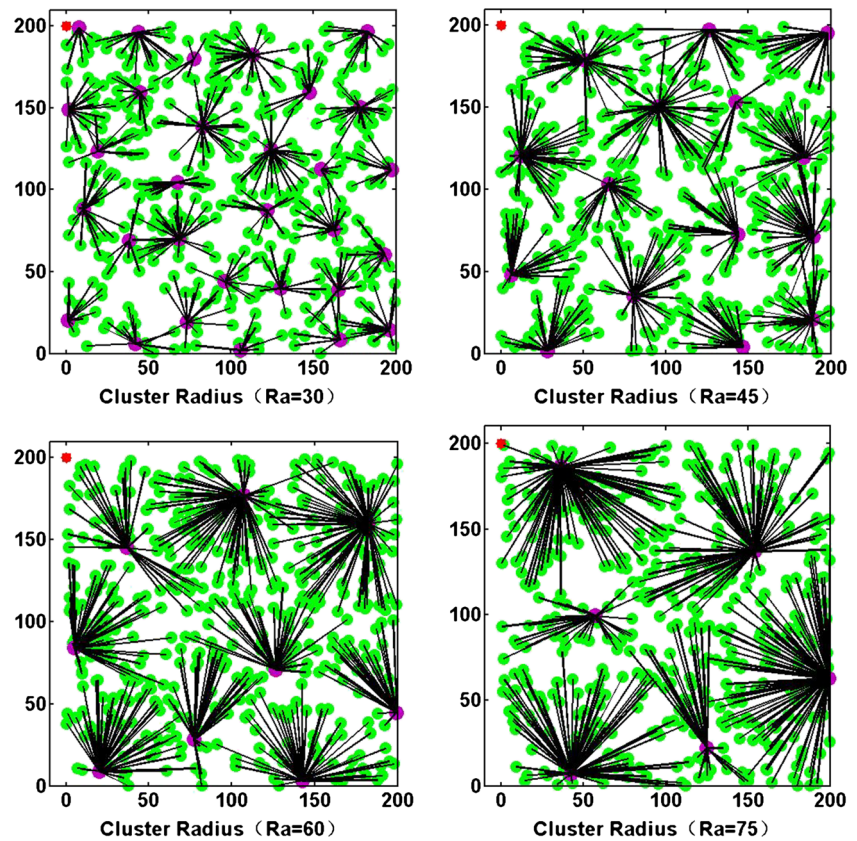


Fig. 7 100–500 node clustering diagram

Fig. 9 The effect drawings of different cluster radii



In addition, the effect drawings of different cluster radius ($R_a = 30, 45, 60, 75$) are shown in Fig. 9.

4.2.2 Comparison of simulation results

A variety of performance metrics are used to compare the performance of protocols in WSNs. In this paper, we use the following metrics:

- *Average energy*: This metric gives the average energy of all nodes at the end of the simulation.
- *Energy consumption*: This metric gives the energy consumption of all the nodes in the network area for sending the data packet to the sink.
- *Standard deviation of energy*: This metric gives the average deviation between the energy levels on all nodes.
- *Network lifetime*: This metric gives the time when the first node exhausts its energy.

By using the simulator developed by MATLAB, we compare the proposed scheme with LECP [24], TEEN [32], MP [33], MACS [34], and MRP [35] protocols. Figures 10, 11, 12, and 13 show simulation results for different network-scale studies. It can be seen that the performance of CRPD is better than the other algorithms in terms of average energy, energy consumption, and standard

deviation of energy, but the performance on the network lifetime is general. However, we can adjust the optimal cluster radius according to the density of the network to achieve the goal of prolonging the network lifetime.

In Fig. 10, the average energy of CRPD and MRP is higher than that of other algorithms, while the average energy of LECP is slightly lower than that of CRPD and MRP. When the number of nodes is less than 300, the performance of CRPD is better than that of MRP. But when the number of nodes is more than 300, the performance of CRPD is slightly lower than that of MRP. This is because with the number of the nodes in the network increasing, the number of cluster member nodes connected to each cluster

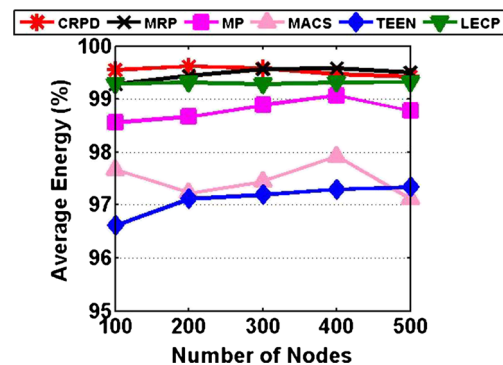


Fig. 10 Average energy

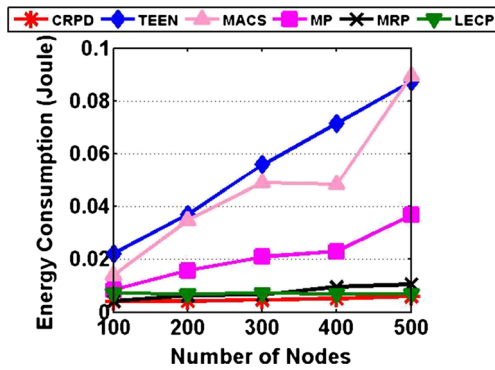


Fig. 11 Energy consumption

head relatively increases, then the energy consumed by the cluster head relatively increases, so the average energy trend of CRPD decreases slightly. The figure shows that there are lots of nodes with more residual energy in CRPD and MRP, which means that CRPD and MRP require less energy to transmit data.

Figure 11 shows the energy consumption of each protocol. The more nodes in the network, the more traffic in the network, which makes TEEN and MACS energy consumption be significantly higher. MACS is one of the most costly algorithms, because it cannot make use of the data correlation and clustering to remove redundant information between neighboring nodes. Since MRP is event-dependent, only nodes around the event consume energy, so its energy consumption is low. Although TEEN is a dynamic clustering algorithm, the structure of cluster in TEEN is not related to the event region. Therefore, the energy consumption of TEEN is higher than that of MRP. And the energy consumption of MP is less than that of TEEN and MACS. The reason is that the main route in the MP is formed according to some metrics, such as low power consumption. Therefore, when source nodes always use the primary route to transmit data, the energy consumption of MP is low. However, the trend of CRPD energy consumption tends to be stable, and it is the lowest and the best one. This is because once clusters are

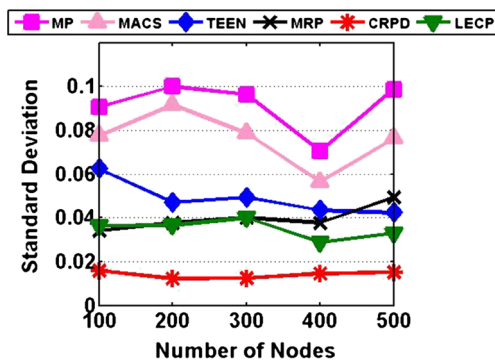


Fig. 12 Standard deviation

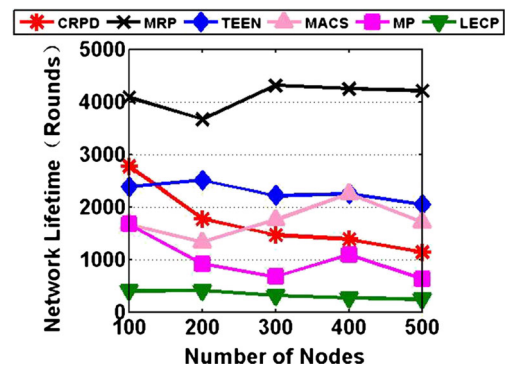


Fig. 13 Network lifetime

constructed in the network, only a few nodes consume energy even if there are nodes joining, moving, or dying, which greatly reduces the energy consumption of the whole network.

In Fig. 12, the standard deviation of CRPD is significantly lower than the other algorithms, which shows that CRPD can effectively balance the energy consumption of all nodes and plays a role in balancing the load.

Figure 13 shows the network lifetime of the six algorithms. It is evident that the network lifetime of MRP is almost twice of that obtained by the other algorithms. The influences of network size on MRP is small. This is because MRP is related to the event area, and only a small number of nodes in the event area consume energy. The network lifetime of CRPD decreases with the increase of network size. However, as the network size increases, the network lifetime of the CRPD decreases. This is because that there are more nodes in the network. More cluster members in a cluster will speed up the death speed of cluster heads, which will reduce the lifetime of CRPD networks. TEEN is better than MACS because it can use clusters to transfer data. The poor performance of MP is because it always adopts the main path to transmit data, which causes the energy of nodes in the main path to be quickly exhausted. Some cluster heads close to the sink in the network are responsible for the intra-cluster communication and the inter-cluster communication; thus, their energy consumption is too fast, which makes LECP performance low.

5 Conclusion

In this paper, a novel energy-efficient routing scheme is proposed for dynamic clustering networks. Through the real-time updating of the network topology, the energy load of the whole network is evenly distributed to each sensor node, which achieves the purpose of load balancing. The routing algorithm uses an acknowledgment mechanism to ensure the reliable delivery of data. In addition, clustering-based data collection reduces the traffic and energy

consumption, and appropriately adjusting the optimal clustering radius can correspondingly extend the network lifetime. Simulation results show that the performance of the proposed protocol is superior to the existing protocols MRP, TEEN, MACS, MP, and LECP in terms of energy efficiency. In the future work, we should further solve the energy efficiency problem of the cluster head node in the communication process, thus further improve the lifetime of the network.

Acknowledgments This work is supported by the NSF of China under grant nos. 61672321, 61771289 and 61373027.

References

1. Stattner E, Vidot N, Hunel P, Collard M (2012) Wireless sensor network for habitat monitoring: a counting heuristic. In: IEEE LCN workshops, pp 853–760
2. Jaigirdar FT, Islam MM (2016) A new cost-effective approach for battlefield surveillance in wireless sensor networks. In: IEEE networking systems and security, pp 1–6
3. Rajaram ML, Kougianos E, Mohanty SP, Sundaravadivel P (2016) A wireless sensor network simulation framework for structural health monitoring in smart cities. In: IEEE, international conference on consumer electronics, pp 78–82
4. Jelacic V, Magno M, Brunelli D, Paci G, Benini L (2013) Context-adaptive multimodal wireless sensor network for energy-efficient gas monitoring. *IEEE Sens J* 13(1):328–338
5. Cheng S, Cai Z, Li J, Gao H (2017) Extracting kernel dataset from big sensory data in wireless sensor networks. *IEEE Trans Knowl Data Eng* 29(4):813–827
6. Li J, Cheng S, Cai Z, Yu J, Wang C, Li Y (2017) Approximate holistic aggregation in wireless sensor networks. *ACM Trans Sens Netw* 11(2):1–24
7. Cheng S, Cai Z, Li J, Gao H (2017) Extracting kernel dataset from big sensory data in wireless sensor networks. *IEEE Trans Knowl Data Eng* 29(4):813–827
8. Zahhad MA, Ahmed SM, Sabor N, Sasaki S (2015) Mobile sink-based adaptive immune energy-efficient clustering protocol for improving the lifetime and stability period of wireless sensor networks. *IEEE Sens J* 15(8):4576–4586
9. Velmani R, Kaarthick B (2015) An efficient cluster-tree based data collection scheme for large mobile wireless sensor networks. *IEEE Sens J* 15(4):2377–2390
10. Shen H, Li Z, Yu L, Qiu C (2014) Efficient data collection for large-scale mobile monitoring applications. *IEEE Trans Parallel Distrib Syst* 25(6):1424–1436
11. Xu C, Liu Z, Xia J, Yu H (2015) An adaptive distributed re-clustering scheme for mobile wireless sensor networks. In: *Wireless communications & signal processing (WCSP)*, pp 1–6
12. Chen L, Xu Z, Liu T, Chen C (2015) A dynamic clustering and routing protocol for multi-hop data collection in wireless sensor networks. In: *IEEE CCC*, pp 7811–7816
13. Jia D, Zhu H, Zou S, Hu P (2016) Dynamic cluster head selection method for wireless sensor network. *IEEE Sens J* 16(8):2746–2754
14. Heinzelman WR, Chandrakasan A, Balakrishnan H (2000) Energy-efficient communication protocol for wireless microsensor networks. In: *International conference on system sciences*, vol 18, p 8020
15. Heinzelman WB, Chandrakasan AP, Balakrishnan H (2002) Application-specific protocol architectures for wireless networks. *IEEE Trans Wirel Commun* 1(4):660–670
16. Kumar D, Aseri T, Patel RB (2009) EEHC: energy efficient heterogeneous clustered scheme for wireless sensor networks. *Comput Commun* 32(4):662–667
17. Bajaber F, Awan I (2011) Adaptive decentralized re-clustering protocol for wireless sensor networks. *J Comput Syst Sci* 77(2):282–292
18. Younis O, Fahmy S (2004) HEED: a hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks. *IEEE Trans Mob Comput* 3(4):366–379
19. Jin Y, Wei D, Vural S, Gluhak A, Moessner KM (2011) A distributed energy-efficient re-clustering solution for wireless sensor networks. In: *Proceedings of the IEEE global telecommunications conference (GLOBECOM)*, pp 1–6
20. Chen YC, Wen CY (2013) Distributed clustering with directional antennas for wireless sensor networks. *IEEE Sens J* 13(6):2166–2180
21. Yu J, Qi Y, Wang G (2011) An energy-driven unequal clustering protocol for heterogeneous wireless sensor networks. *J Control Theory Appl* 9(1):133–139
22. Yu J, Qi Y, Wang G, Guo Q, Gu X (2011) An energy-aware distributed unequal clustering protocol for wireless sensor networks. *Int J Distrib Sens Netw* 2011(3):876–879
23. Yu J, Qi Y, Wang G, Gu X (2012) A cluster-based routing protocol for wireless sensor networks with nonuniform node distribution. *AEÜ—Int J Electron Commun* 66(1):54–61
24. Yu J, Feng L, Jia L, Gu X, Yu D (2014) A local energy consumption prediction-based clustering protocol for wireless sensor networks. *Sensors* 14(12):23017–40
25. Jin Y, Wang L, Kim Y, Yang X (2008) EEMC: an energy-efficient multi-level clustering algorithm for large-scale wireless sensor networks. *Comput Telecommun Netw* 52(3):542–562
26. Almalkawi I, Zapata M, AlKaraki J (2012) A cross-layer-based clustered multipath routing with QoS-aware scheduling for wireless multimedia sensor networks. *Int J Distrib Sens Netw* 2012(12):184–195
27. Yang W, Fu Z, Kim J, Park M (2007) An adaptive dynamic cluster-based protocol for target tracking in wireless sensor networks. In: *APWeb/WAIM*, pp 157–167
28. Wang F, Bai X, Guo B, Liu C (2016) Dynamic clustering in wireless sensor network for target tracking based on the fisher information of modified Kalman filter. In: *Systems and Informatics (ICSAI)*, pp 696–700
29. Sharma S, Jena SK (2015) Cluster based multipath routing protocol for wireless sensor networks. *Comput Sci* 45(2):14–20
30. Shi Q, Huo H, Fang T, Li D (2009) A 3D node localization scheme for wireless sensor networks. *IEICE Electron Express* 6(3):167–172
31. Chuang PJ, Jiang YJ (2014) Effective neural network-based node localisation scheme for wireless sensor networks. *IET Wirel Sens Syst* 4(2):97–103
32. Manjeshwar A, Agrawal DP (2001) TEEN: a routing protocol for enhanced efficiency in wireless sensor networks. In: *Proceedings of IPDPS Workshops*, pp 2009–2015
33. Ganesan D, Govindan R, Shenker S, Estrin D (2001) Highly-resilient, energy-efficient multipath routing in wireless sensor networks. *ACM Sigmobile Mobile Comput Commun Rev* 5(4):11–25
34. Ren X, Liang H, Wang Y (2008) Multipath routing based on ant colony system in wireless sensor networks. *Comput Sci Softw Eng* 3:202–205
35. Yang J, Xu M, Zhao W, Xu B (2010) A multipath routing protocol based on clustering and ant colony optimization for wireless sensor networks. *Sensors* 10(5):4521–40