Research Article

Bee-Sensor-C: An Energy-Efficient and Scalable Multipath Routing Protocol for Wireless Sensor Networks

Xuelian Cai, Yulong Duan, Ying He, Jin Yang, and Changle Li

State Key Laboratory of Integrated Services Networks, Xidian University, Xi’an, Shaanxi 710071, China

Correspondence should be addressed to Changle Li; clli@mail.xidian.edu.cn

Received 20 March 2014; Revised 19 August 2014; Accepted 24 August 2014

Academic Editor: Li-Hsing Yen

Copyright © 2015 Xuelian Cai et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

A wireless sensor network (WSN) is composed of a large collection of sensor nodes with limited resources in terms of battery supplied energy, processing capability, and storage. Therefore, the design of an energy-efficient and scalable routing protocol is a crucial concern for WSN applications. In this paper, we propose Bee-Sensor-C, an energy-aware and scalable multipath routing protocol based on dynamic cluster and foraging behavior of a bee swarm. Bee-Sensor-C is an evolution from BeeSensor which is a bee-inspired routing protocol for WSNs. First of all, through introducing a dynamic clustering scheme, Bee-Sensor-C offers parallel data transmissions close to the event area. This evolution reduces routing overhead and improves the scalability. Moreover, Bee-Sensor-C adopts an enhanced multipath construction method in order to achieve the balance of the network energy consumption. Besides, Bee-Sensor-C can well support the multicluster scenario. Through simulations, the network performance is evaluated and the results demonstrate that Bee-Sensor-C outperforms the existing protocols in terms of energy efficiency, energy consumption balance, packet delivery rate, and scalability.

1. Introduction

Wireless sensor networks (WSNs) consisting of multiple collaborative sensor nodes have been widely applied in the field of medical, industrial, and military applications [1, 2]. In these applications, tiny sensors equipped with embedded processors, memory, and radio transceivers are randomly deployed in a target area to form a network that realizes event detection, information gathering, and communication. WSNs can provide users with the ability to record, observe, and react to an event or a phenomenon in a specific environment. And WSNs are expected to play an even more important role in the next generation network to sense the physical world [3, 4].

WSNs have many characteristics that differ from conventional wireless networks [5]. The sensor nodes in WSNs typically have limited resources in terms of battery supplied energy, processing capability, communication bandwidth, and storage [6]. Moreover, due to the varying operating environment and requirements of WSNs, issues, including large scale, dynamic topology, and autonomous, distributed operation of the node, should also be carefully taken into account.

Routing is a very important function to achieve efficient communication in the design of WSNs [7] and has been shown to significantly impact the energy efficiency and consequently the lifetime of WSNs [8]. The sensor nodes in WSNs sense the surrounding environment and deliver the measured data to a central base station or the sink through the routing process. Nevertheless, the design of routing protocols in WSNs is a challenging task as a result of their inherent characteristics mentioned above. An applied routing protocol in WSNs should ensure the minimum of the energy consumption and hence maximization of the lifetime of the network [9]. That means that it not only aims for energy efficiency but also aims for energy consumption balance. In addition, an ideal routing protocol should be scalable and resilient to meet the requirements of WSNs.

Recent research has proven that both multipath and clustering communication are very efficient routing methods in WSNs. Clustering has been widely used to extend the network lifetime and achieve network scalability. Data from different
sources in cluster are aggregated by reducing redundancy with the purpose of minimizing energy consumption in transmission. On the other hand, in multipath mechanism, two or more paths are established from source to destination [10]. Multipath routing can distribute the forwarding of the data packets across multiple paths so that all nodes can utilize their batteries at a comparable rate, which contributes to prolonging the network lifetime and achieving load balance.

Swarm intelligence (SI) has been considered in the routing issue of an WSN [11]. Social insect communities such as ants and honey bees have many desirable properties in the WSN perspective as surveyed in [7, 12]. These communities are formed with simple, autonomous, and cooperative organisms that are interdependent for their survival [6]. Despite the lack of centralized control, social insect communities are able to effectively coordinate themselves to achieve global objectives. Their labor division is clear and can be adaptively adjusted according to the colony requirements or environment changes in an unpredictable world. The characteristics described above are necessary and desired in the context of sensor networks, especially for the design of routing protocols in WSNs. Bioinspired mechanisms are considered more robust as they provide interesting solution for routing due to their inherent features [10].

Honey bee colonies have recently emerged as a source of inspiration for the optimization of time-varying, dynamic, and multiobjective problems [13]. Some researchers have paid much attention to the foraging behavior of bee swarm and utilize it in designing the routing protocol in last decade. Such systems may be composed of simple nodes working together to deliver messages, while being resilient against changes in its environment. Nevertheless, to the best of our knowledge, limited number of bee-inspired routing protocols for WSNs have been reported in [13–15]. Moreover, these existing works have not considered the integration of clustering technique, multipath mechanism, and bee-inspired mechanism in solving routing problem for WSNs.

In this paper, we propose an energy-efficient and scalable multipath routing protocol for WSNs, Bee-Sensor-C by integrating a dynamic clustering scheme and enhanced BeeSensor proposed in [13]. In our protocol, bee agents were modeled to suit the energy resource constraints in WSNs for the purpose of constructing cluster near the event source and finding the multiple paths of better quality, by extensively borrowing from the principles behind the honey bee communication. The Bee-Sensor-C’s design considers several restrictions in respect to resource availability and scalability for WSN. And, furthermore, Bee-Sensor-C has four important features which are listed as follows.

(i) Bioinspired technique: Bee-Sensor-C is based on bee-inspired mechanism.

(ii) Dynamic clustering scheme: Bee-Sensor-C adopts a dynamic clustering scheme to provide parallel data transmission near the event.

(iii) Multiple paths selection: Bee-Sensor-C adopts an enhanced multipath construction method to achieve the energy consumption balance.

(iv) Multicluster scenario support: Bee-Sensor-C takes the multicluster scenario into consideration based on the conclusion of our mathematical analysis.

By our experimental evaluation, Bee-Sensor-C performs better in restricting routing overhead and energy efficiency than other SI based WSN routing protocols in our scenarios. What is more, Bee-Sensor-C is scalable and can significantly balance energy consumption among the nodes in the event-based application scenario.

The rest of this paper is organized as follows. In Section 2, we briefly introduce some related routing algorithms. Section 3 presents a brief description of the social insect analogy. The details of the Bee-Sensor-C algorithm are described in Section 4. In Section 5, we describe the evaluation and simulation results. Finally, we conclude the paper together with outlook for future work.

2. Related Works

We now briefly present some well-known routing protocols that are based on cluster or multipath for WSNs. Subsequently, we describe some swarm intelligence based routing protocols mainly taking inspiration from the foraging behaviors of ant or bee colonies.

2.1. Cluster-Based and Multipath Routing Protocols.

In the last few years, a relatively large number of clustering routing protocols have been developed for WSNs. LEACH [16] is a representative clustering algorithm for WSN. LEACH forms clusters by using a distributed algorithm. However, it is not friendly in a large network deployment. The authors in [17] design a power-efficient and adaptive clustering hierarchy protocol (PEACH). In PEACH, cluster formation is performed by using overhearing characteristics of wireless communication to avoid additional overheads. BEE-C in [14] based on LEACH-C algorithm uses a bee-inspired algorithm (honey bee mating optimization, HBMO) to perform the clustering of network nodes. Since it is applied to networks with continuous data dissemination, the main objective of BEE-C is to extend the network lifetime by reducing the energy consumption. However, it is poor in scalability due to single hop communication between the cluster head and base station.

Multipath routing protocols can extend the network lifetime because they favor battery depletion at a comparable rate. In [18], the authors propose an energy-efficient multipath routing protocol (EEMRP) for WSNs. EEMRP utilizes a distributed multipath search algorithm to discover multiple paths and considers the remaining energy and hop counts as evaluation index of the link cost function. However, the reliability of successful paths is poor in EEMRP. The authors in [19] present a multipath energy-aware routing protocol (MERP) for WSNs. MERP uses only the local information to find multiple paths between a pair of source and sink node and designs a reliable data transmission algorithm based on an order of the quality of paths. However, MERP is not suitable for large scale WSNs because of the great storage costs of each node. In [20], the authors combine the advantages of
multipath routing, enhanced overlay scheme, and clustering technique to make network more energy-efficient and fault tolerant and to get more robust data delivery. According to their simulations, the proposed approach is more energy efficient and robust in terms of data delivery rate than other multipath routing protocols.

2.2. Swarm Intelligence Based Routing Protocols. The idea of using ant-inspired algorithm to establish routes in WSNs is not new. A lot of ant-inspired algorithms for WSNs have been proposed. In FF-Ant (flood forward ant routing) [21], the source nodes flood the forward ants stochastically towards the sink node. If the search is successful, forward ants will create backward ants to traverse back to the source. Multiple paths are updated per flooding. Besides, two methods are used to restrict the flooding process. However, the authors only focus on the building of an initial pheromone distribution, which is not suitable for the high-density networks. The energy efficient ant-based routing algorithm (EEABR) proposed in [22] is designed to extend network lifetime by reducing the communication overhead in path discovering. In EEABR, each node in the network launches a forward ant with fixed size at a regular interval for finding a route to the destination. When the forward ant reaches its destination, it is converted to backward ant with the mission to update the pheromone trail of the path. EEABR performs better in terms of energy efficiency, standard deviation in the battery levels, and minimum remaining energy of a node. The authors in [23] propose three improvements to the EEABR algorithm to further improve its energy efficiency. These improvements are based on a new scheme to intelligently initialize the routing tables, intelligent update of routing tables in case of a node or link failure, and reduce the flooding of ants for congestion control.

Foraging strategy of honey bees has been extensively studied but their utilization in the design of routing protocols is relatively new, especially in WSNs. Saleem et al. in [13] propose an energy-efficient and scalable routing protocol for WSNs, BeeSensor, by taking inspiration from relevant features of BeeAdHoc [24] that is primarily designed for MANET. Besides, in [25], they present and analyze the algorithm in more detail. Like BeeAdHoc, BeeSensor works with four types of agents: packers, scouts, foragers, and swarms. These agents live in a software module called bee-hive for each sensor and undertake different types of tasks. Packers receive data packets from the upper layer and look for suitable foragers. Scouts are used to discover new routes and are divided into two sorts: forward scouts and backward scouts. Foragers carry data packets to the destinations along the unique path. BeeSensor can achieve good performance in energy efficiency by restricting the number of control messages. Moreover, it is scalable with the less memory and processing costs.

Although the researchers above have been engaged in a series of bee-inspired algorithms, they have not proposed a suitable clustering idea for large-scale WSNs. Moreover, these algorithms have a poor performance in the application scenario where the nodes near an event need parallel data transmission. In order to improve the performance, we proposed an energy-efficient and scalable routing protocol, Bee-Sensor-C, by combining a dynamic clustering technique, multipath scheme, and bee-inspired routing.

3. The Social Insect Abilities

In nature, scout bees explore the surrounding of the hive in order to detect possible source of food. Once a flower patch is discovered, the scout bee returns back to the hive to recruit the forager bees through a special dance. The dance is named “waggle dance” and determined fundamentally by the quality, distance, quantity, and direction of food source found. According to “waggle dance,” some bees are recruited and become foragers to collect the food. Furthermore, when a forager successfully returns back to the hive each time, it can also preform the foraging dance and report any improvement or deterioration of the currently working path. If the path’s reliability becomes very poor, the forager will stop recruiting new members. In the following, we briefly discuss three key characteristics of the bee colony that can be exploited to solve the routing problems in wireless communication networks.

(1) Management of Work Division. Labor division of bee colony is clear and can be adjusted according to the colony requirements or environment changes. Although all bees are uniform individuals, each bee can play a different role and change in accordance with the colony requirements. For instance, foragers are mainly workers to exploit nectar sites. However, they may also give up the nectar site being currently exploited and switch to scout bees to search for an alternative site. This is an important characteristic of bee agents which is the colony’s adaptive behavior to environment changes.

(2) Multiple Interactions. The bees communicate and exchange information with each other through the so-called dance [26]. For instance, the scout returns back to hive and recruits the foragers through “waggle dance.”

(3) Stochastic Flower-Site Selection. A colony always tries to explore several food sites instead of the best one. Moreover, the number of foragers is proportional to the quantity of food source. The foragers stochastically choose a foraging site. Therefore, the better the quality of flower site is, the more foragers there will be.

These strategies of foraging process implemented by the bee colony have desired properties of routing protocols in WSNs and can be utilized in the design of the routing protocol. In the following section, we will introduce our protocols based on the foraging principles of honey bees.

4. Bee-Sensor-C

4.1. Protocol Description. Bee-Sensor-C is an event driven and on-demand multipath routing protocol for WSNs. As shown in Figure 1, Bee-Sensor-C is mainly divided into three phases: cluster formation, multipath construction, and data transmission. Assume that the entire network has been initialized completely and each node has its own ID and relative
sink node ID before the cluster formation. The first phase is to build cluster structure when an event happens. In the second phase, we make some improvements to BeeSensor for constructing multipath between CH and relative sink, followed by carrying data to sink through the stochastically selected path. At last, we realize the route maintenance.

4.1.1. First Phase: Cluster Formation. When an event occurs in the network, all nodes near the event may need to send the perceived data. In this situation, BeeSensor is poor in the overall performance. Therefore Bee-Sensor-C adopts an event-based dynamic clustering algorithm. Being different from BeeSensor, Bee-Sensor-C adds a new agent called HiveHeader into bee-hive for each sensor. The major responsibility of HiveHeader is to claim that the node wants to be a cluster header (CH) node in an event area when it detects an event.

When the event in the network happens, nodes nearby will become activated and measure the specific perceived attribute. And then the nodes having information about the event will join cluster. The clustering region is called event-sub-network and the region outside the event-sub-network is referred to as outside-network.

The clustering algorithm obeys the following rule: node $i$ can decide whether to join the cluster according to the received signal strength (RSS) of relative event. that is, if $RSS_i \geq RSS_{\text{threshold}}$, node $i$ will be located in the event-sub-network. Furthermore, the size of cluster can be adjusted dynamically according to the threshold value of RSS ($RSS_{\text{Threshold}}$).

Once the cluster forms, Bee-Sensor-C takes an efficient mechanism, the first declaration win rule, to select CH node. The principle can be stated in detail as follows.

Each node has a HiveHeader agent. When node $i$ detects an event, it will wait $T_i$ before broadcasting its HiveHeader agent in the event-sub-network ($T_i$ is the waiting time). HiveHeader of node $i$ will be replaced by that of node $j$ when it receives HiveHeader of node $j$. If node $i$ receives different HiveHeaders from the different nodes simultaneously, it will choose the node with higher residual energy. But this situation hardly occurs. Finally, the node that firstly sends HiveHeader will be the only CH node, and the other nodes become cluster members in the event-sub-network. $T_i$ is calculated using

$$T_i = t \cdot \frac{\alpha - (\text{RadioRange} - d_i)}{\beta \times Er_i},$$

(1)

where $t$ is a constant time to control the value of $T_i$ and can be adjusted appropriately based on specific network conditions. Usually, the value of $t$ is equal to the time that the HiveHeader can travel once in the event area. $Er_i$ is the current residual energy of node $i$. RadioRange is the maximum radio range of node. $d_i$ is the distance from event source to node $i$. $\alpha$ and $\beta$ are two user-defined constants which can be adjusted so that different values of $T_i$ can differ approximately by $t$.

Because the energy consumption is relative to the distance between two nodes, we can find that the sum of energy consumption is the least when the node located in the center of an area communicates with other nodes in the area. Therefore, we make $T_i$ proportional to $d_i$, as shown in (1).

Equation (1) shows that a node that is with higher residual energy and closer to the event source will send HiveHeader earlier and have the most opportunity of becoming a CH in the event-sub-network. Apart from the agent type field, the HiveHeader also carries four additional information fields in the header: event ID, source node ID ($S_{\text{ID}}$), waiting time ($T_{\text{wait}}$, i.e., $T_i$), and the current remaining energy of the node ($Er$). The packet format of HiveHeader is shown in Figure 2.

Each cluster has its TTL (time to live). CH gathers and aggregates the data from the member nodes in the event-sub-network. After that, it will send the data to the sink node until the event stops. When the remaining energy of current CH is lower than 60% (the value can be adjusted according to practical situation), the average energy of all nodes in
the event-sub-network, CH will be reselected by the method above.

4.1.2. Improvements to BeeSensor for Multipath Construction. When CH needs sending data to sink node, it looks whether there are appropriate foragers of any nodes in the event-sub-network for the existing multipath. If CH fails to find valid foragers, an improved BeeSensor algorithm is used to establish multipath between CH and a relative sink node.

Like BeeSensor, Bee-Sensor-C utilizes forward scouts to explore the outside-network in search of an interested sink node using a broadcasting principle. Firstly, CH broadcasts a forward scout to its neighbors in outside-network. However, there is no limit to the hop of forward scouts in BeeSensor, and this will result in excess communication overhead. Therefore, we make the first improvement that realizes the self-destruction of forward scouts. The method is to add an allowable maximum value of the hops between CH and sink node into forward scouts. We define the maximum value as $H_{\text{max}}$ which is a tradeoff between the number of paths and communication overhead and can be estimated. When the number of hops is higher than $H_{\text{max}}$, self-destruction of forward scouts occurs. We redefine the packet format of the forward scout shown in Figure 3, where $D_{\text{ID}}$ is the destination node ID, $E_{\text{min}}$ is the minimum residual energy of nodes in the path, $H$ is the hop count until the current node, and $E_{\text{avg}}$ is the average residual energy of nodes until the current node.

In BeeSensor, when an intermediate node $i$ receives a forward scout form node $j$, it computes a reward value $R_{ji}$ given by $R_{ji} = E_{ji}/H_{ji}$, where $E_{ji}$ is the minimum remaining energy and $H_{ji}$ is the hop count of the path until the current node. Moreover, node $i$ stores $R_{ji}$, the source node ID, the scout ID, and the previous hop in a cache table. For constructing multipath, the intermediate node should obey the following rules.

Entries that consisted of reward value and previous hop in the cache should be updated by future forward scouts only when the new reward value is greater than that existing in the scout cache. Furthermore, the future forward scouts are only utilized to update the cache and will be dropped instead of being rebroadcasted. Upon reaching the sink, the forward scout will be converted into backward scout with a unique path ID. Then, the backward scout travels back to the CH node along the backward route according to the previous hop information in the cache. Each intermediate node creates a forwarding table with path ID, the next hop, and previous hop. For providing loop freedom, a backward scout will be dropped by the node that received the backward scout for the second time, and the corresponding entry is flushed.

However, we find some questions about the above process of BeeSensor in our simulations. Firstly, we find that the variation of reward is not reasonable as the hop count increases. Moreover, the reward does not take into consideration the average residual energy along the path. Consequently, we make the improvement to reward that is calculated according to the following equation:

$$R_{ji} = \omega_1 E_{\text{avg}} + (1 - \omega_1) E_{ji}/H_{ji}, \quad (2)$$

where $E_{\text{avg}}$ is average remaining energy of the path and $\omega_1$ is the control parameter.

In Bee-Sensor-C, if the residual energy of node $i$ is far less than the average residual energy of the path (we define the node as “abnormal node”), it will directly drop the forward scout and not do further processing in order to prolong the network lifetime.

Secondly, when a lot of nodes are randomly distributed in a target area, there may be dense nodes in a small region. In this situation, it is possible that only one node can be the member of multiple paths even though other nodes are supposed to be potential member of paths. Therefore, the method of updating the cache/reward in BeeSensor is not conductive to multipath construction in some cases. For instance, in Figure 4, the nodes in area 1 can communicate with nodes in area 2. When node 3 and node 4 receive the forward scout from node 2 and node 1, they update the cache and consider node 1 as the previous node because the reward from node 1 is of higher value. For the same reason, other nodes in area 2 all choose node 1 as the previous hop. The reward of node 2 is very close to that of node 1, although the former is less than the latter. Thus node 2 is not able to become the intermediate node of any path. In this case, there are some potential paths that are dropped.

Based on the above-mentioned consideration, we make the improvement to cache update as follows.

We assume that $R_1$ is the reward along path 1, which is stored in node 3. And $R_2$ is the newly received reward along path 2. When $R_2 \gg R_1$, node 3 updates the reward value and the previous hop entry in the cache. And when $R_2 \ll R_1$, node 3 does not update the cache. Only when $R_1$ and $R_2$ meet the following formula:

$$\left| \frac{R_1 - R_2}{R_1} \right| \leq \delta, \quad (3)$$
where $\delta$ is the relative difference, and generally equal to 0.05, the cache will be updated with the probability $p$ given by

$$
p = \begin{cases} 
\left( \frac{R_2}{R_1 + R_2} \right)^2 & (R_2 \leq R_1) \\
\frac{R_2}{R_1 + R_2} & (R_2 > R_1)
\end{cases}.
$$  

(4)

Using the method above, some potential paths may be built. As shown in Figure 4, path 1 along node 3 may be established. This contributes to the balancing of the energy consumption and the prolonging of the lifetime of network.

At last, we find an interesting phenomenon that the node which is the neighbor of sink may choose other neighbor of sink as the previous hop. This reduces the number of paths and results in quick fade of some nodes near the sink. As shown in Figure 5, node D receives the forward scout from node B and node C, it choose node B as the previous hop due to the node B’s higher reward. The path covering node C may pass through relatively more hops, which leads to lower reward. When the backward scout travels back, the potential path which passes sink, node D, node B and node A in sequence will be dropped. Moreover, the other neighbor nodes of sink may encounter the same situation, especially when the source node is far away from sink.

In order to solve the problem above, we proposed the method that the node will directly send the forward scout to sink instead of broadcasting only when the node is the neighbor of sink. Of course, all the nodes in the network know whether they are the neighbors of sink or not.

4.1.3. Data Transmission. In Bee-Sensor-C, when the backward scout with unique path ID arrives at the CH, it recruits foragers using the "waggle dance" of honey bees. The dance number represents the quality of the path and is computed using

$$
\text{DN}_{pid} = \left[ \frac{\eta - H_{pid} \times (\omega_2 \cdot (E_{\text{initial}} - E_{\text{avg}}) + (1 - \omega_2) \cdot E_{\text{min}})}{\lambda} \right],
$$  

(5)

where $\omega_2$ is the weight value, $\eta$ and $\lambda$ are user-defined constants, $E_{\text{initial}}$ is the initial energy of node, $E_{\text{min}}$ is the minimum residual energy in the path, $E_{\text{avg}}$ is the average residual energy of nodes in the path, and $H_{pid}$ is the hop count of the path.

Once multiple paths between CH and relative sink are constructed, the CH will send the data packets to sink node using foragers that represent valid routes to the sink node. The probability of selecting path $i$ is computed as follows:

$$
P_i = \frac{\text{DN}_i}{\sum_{j=1}^{n} \text{DN}_j},
$$  

(6)

where $\text{DN}_i$ is the number of distance of path $i$, $n$ is the set of available paths. Since the path is selected probabilistically, load balance that maximizes the network lifetime can be achieved.

4.1.4. Repair Strategy for Situation of Unreachable Sink. In Bee-Sensor-C, there may be a circumstance that all forward scouts could not reach the destination before their self-destruction, which results from route discovery scheme. We have to add a repair strategy to cope with this situation.

As shown in the route discovery scheme, the forward scout is restricted by the hop limit and its transmission probability. The former search failure means that forward scouts need more opportunities to search further in order to access the sink.

At the very beginning, the forward scouts set the initial hop limit and the CH sets a timer waiting for time-out. The hop limit will double its current value after a search failure and will maintain the value when it reaches the maximum hop limit, which is predefined according to the network scale. Each time the hop limit is set, the time-out will be correspondingly modified.

4.1.5. Route Maintenance. In Bee-Sensor-C, route maintenance is very necessary and obeys the rules as follows.

In order to detect the validity of path, the last forager arriving at the sink within a certain time will be returned towards the CH using swarm. Swarm has two functions, one is to determine the validity and the other is to collect and update the information about the quality of path.

If none of swarms come back within a specified time period, we think the path becomes invalid. The relative information in routing table and forwarding table will be deleted. Moreover the agents in invalid path will execute self-destruction. When the number of multiple paths is less than two, the CH will start a new route discovery process.

Each event-sub-network has its TTL. When the cluster is dead, relative information will be removed.

4.2. Mathematical Modeling of Key Performance Metrics. For the purpose of finding out the key factor affecting the performance of our protocol, we make the following mathematical analysis mainly on two performance metrics: routing overhead and energy consumption.
4.2.1. System Model. The WSN consists of N nodes which are static and uniformly distributed on a square field using a homogeneous Poisson distribution with node density ρ. The communication is symmetric, and two nodes within the transmission radius r₀ are neighbors in the network. To ensure the network connectivity, we assume that the number of neighbors of node, defined as node degree d(i), is satisfied with the conditions in [27]. N nodes in the event-sub-network are grouped into clusters. In order to evaluate performance under ideal routing condition, we assume perfect MAC conditions and link quality. Furthermore, we use the radio model as follows:

\[ E_T = E_{elec} + E_{amp} \times d^2 \]
\[ E_R = E_{elec} \]

where \( E_T \) and \( E_R \) are the energy consumption of transmitting and receiving a 1-bit message, respectively. \( E_{elec} \) is the energy consumed by the electronics device for transmitting and receiving a 1-bit message, and \( E_{amp} \) is the transmission amplification energy.

Then, we analyze the routing overhead and energy consumption of Bee-Sensor-C based on the models in [28].

4.2.2. Routing Overhead. We define routing overhead (Cp) as follows: the number of control packets needed before the multipath has been set up. It consists of the number of HiveHeader (Cp₁) during the cluster formation phase, the number of forward scouts (Cp₂), and the number of backward scouts (Cp₃) during the second phase. We have

\[ Cp = Cp_1 + Cp_2 + Cp_3. \] (8)

Bee-Sensor-C adopts an event-based dynamic clustering algorithm, and the overhead of this phase is given by

\[ Cp_1 = (N_C - 1) \times 2 = (\pi \cdot d_{thr}^2 \times \rho - 1) \times 2; \] (9)

where \( d_{thr} \) is the radius of event-sub-network. In the path discovery phase, we have

\[ Cp_2 = (d_{avg} - N_C) \times P_s + \left( (d_{avg} - N_C) \cdot P_s \right) \times P_s \times d_f [1] \]
\[ + \cdots + P_s \cdot \left( (d_{avg} - N_C) \cdot P_s^{h-1} \cdot d_f [1] \cdots d_f [h-2] \right) \]
\[ \cdot d_f [h-1] \]
\[ = (d_{avg} - N_C) \times P_s \]
\[ + (d_{avg} - N_C) \sum_{i=1}^{h-1} P_s^{i+1} \prod_{j=1}^{i} d_f [j] \quad (h \leq H_{max}), \] (10)

where \( d_{avg} \) is the average value of node degree, \( d_f [j] \) is the expected forward degree of nodes at \( j \) hops from CH defined in [28], and \( P_s \) is the probability that a node will forward a forward scout to its neighbors and the forward scout will be successfully received. Based on our assumption, we have \( P_s = 1 \).

When the backward scouts come back, the overhead is given by

\[ Cp_3 = L_a \cdot m, \] (11)

where \( L_a \) is the average length of multiple paths and \( m \) is the number of available paths.

Combining (8), (9), (10), and (11), the overhead of Bee-Sensor-C is given by

\[ Cp^{(b-s-c)} = (d_{avg} - N_C) \times P_s + \left( (d_{avg} - N_C) \sum_{i=1}^{h-1} P_s^{i+1} \prod_{j=1}^{i} d_f [j] \right) \]
\[ + L_a \cdot m + \left( \pi \cdot d_{thr}^2 \times \rho - 1 \right) \times 2 \quad (h \leq H_{max}). \] (12)

In the same scenario, using the same method, we can get the routing overhead of BeeSensor using

\[ Cp^{(b-s)} = N_C \times \left[ d_{avg} \times P_s + \sum_{i=1}^{h-1} P_s^{i+1} \prod_{j=1}^{i} d_f [j] \right] \]
\[ + P_b \sum_{i=H_l}^{H_l-1} P_b^{i+1} \prod_{j=H_l}^{i} d_f [j] + L_a \cdot m, \] (13)

where \( P_b \) is a given probability. Comparing (12) to (13), we can find that the overhead of Bee-Sensor-C is much lower than that of BeeSensor. In particular, the overhead will decrease when there exist some abnormal nodes in the path.

4.2.3. Energy Consumption. We define energy consumption (\( G_{total} \)) as the total energy consumption in the route establishment and data transmission. It consists of the energy consumption of cluster (\( G_{CH} \)), the energy consumption in the second phase (\( G_{bd} \)), and the energy consumption of data transmission (\( G_{data} \)). Therefore, we have

\[ G_{total} = G_{CH} + G_{bd} + G_{data}. \] (14)

In the event-sub-network, we assume the convergence degree of cluster is \( \alpha \) and the energy consumption of convergence/bit is \( E_c \). In the first phase, Bee-Sensor-C adopts an event-based dynamic clustering algorithm. We have

\[ G_{CH} = Cp_1 \times B_{HiveHeader} \times (E_T + E_R) \]
\[ + (E_c + E_T + E_R) \times B_d, \] (15)

where \( B_{HiveHeader} \) is the size of HiveHeader and \( B_d \) is the total size of data packet. And then

\[ G_{bd} = Cp_2 \times B_{scout} \times (E_T + E_R) \]
\[ + Cp_3 \times B_{scout} \times (E_T + E_R), \] (16)
where \( B_{\text{scout}} \) is the size of scout, and we assume \( B_{\text{HiveHeader}} = B_{\text{scout}} = B_{\text{agent}} \). When the data is transmitted along the multiple paths, the energy consumption is given by
\[
G_{\text{data}} = L_a \times \alpha \times B_d \times (E_T + E_R).
\] (17)

Therefore, we have
\[
G_{\text{total}}^{(b-s-c)} = \left( C_p^{(b-s-c)} \times B_{\text{agent}} + (L_a + \alpha + 1) \times B_d \right) \\
\times (E_T + E_R) + B_d \times E_c.
\] (18)

Using the similar method, the energy consumption of BeeSensor is given by
\[
G_{\text{total}}^{(b-s)} = \left( C_p^{(b-s)} \times B_{\text{agent}} + (L_a + 1) \times B_d \right) \\
\times (E_T + E_R).
\] (19)

Because the energy consumption of convergence is relatively less than the others, a comparison of (18) and (19) proves that Bee-Sensor-C is better in energy efficiency in our suggested scenario.

Also, we can cope with the routing overhead in another aspect. Taking the data amount of intermediate node as the target optimal parameters, we can prove that our method is better in routing discovery. Now, we separate the proof into two parts.

1. When intermediate node receives the packet formerly transmitted by this node, it will conventionally drop it and do nothing. Node in our scheme will also drop it as the His is smaller than the cache. So, performance of the normal method is the same as that of our method.

2. When intermediate node receives the packet for the first time, it will conventionally broadcast the packet. While our method will broadcast the packet depending on the His value, that means the performance of normal method is worse than that of our method.

According to the mathematical analysis, we achieve the following conclusion.

- Routing overhead is the main source of energy consumption.
- Reducing the total number of nodes that participate in broadcasting forward scouts can effectively reduce routing overhead.
- Although the cluster construction and maintenance would bring about more complexity and energy consumption, it is well worthwhile integrating dynamic cluster structure.

Then, we will discuss the scenario that more than one cluster is created in the WSN based on the conclusion above.

### 4.3. Special Discussion on Multiple Clusters

#### 4.3.1. Cluster Reuse.

Each cluster has its TTL, and each forwarding table entry at intermediate nodes in the paths likewise has an associated lifetime. If the event in the event-sub-network stops (i.e., no relative data in the cluster are transmitted), the cluster will also maintain a period of time. When a cluster is alive, other clusters can utilize its information. Moreover, CH belonging to living cluster can reply the route discovery with a backward scout, and the new cluster can construct the path along the CH of the living cluster.

For instance, in Figure 6, cluster 1 is old and alive. When cluster 2 needs to build the multipath between CH 2 and sink, CH 2 broadcasts a forward scout for route discovery. Once the nodes except CH 1 in the cluster 1 receive the forward scout, they will send it to CH 1 directly. When the CH 1 receives the forward scout, it will reply a backward scout to CH 2 along path 1 instead of broadcasting it. The back scout includes the information about the quality of paths between CH 1 and sink.

The relative values, such as \( E_{\text{avg}} \) and \( E_{\text{min}} \), are calculated by the weighted method. For example, \( E_{\text{avg}} \) is given by
\[
E_{\text{avg}} = \sum_{i=1}^{N} p_i E_{\text{avg}},
\] (20)

where \( N \) is the number of available paths between CH 1 and sink, \( p_i \) is the probability of selecting path \( i \), and \( E_{\text{avg}} \) is the average residual energy of each path between CH 1 and sink. Moreover, the nodes in available paths between CH 1 and sink will also drop the forward scout from other nodes, such as the nodes in path 2, which can provide loop freedom. Once the multipath between CH 2 and sink is constructed, CH 2 will send data along these paths. When CH 1 receives the data from CH 2, it will choose the next hop with probability \( p_i \).

#### 4.3.2. Number of Signals That Can Be Received Simultaneously.

If there are multiple events that occurred almost simultaneously in the environment, plenty of clusters will be created. Obviously, here is a question that how many signals can be transmitted to sink. In the following part, we give a mathematical analysis on the number of signals that can be received from the configuration.

As we can see in Figure 7, two clusters have overlap area; nodes in such area will sense both two events. Assuming...
that event 1 and event 2 occurred almost simultaneously and cluster headers are not in overlap area, each node, in overlap area, such as node \( x \), becomes a member of both clusters. Node \( x \) will have to gather two kinds of information and process both of them, which has put great pressure on node \( x \). In the other case that node in overlap area (i.e., node \( x \)) becomes the header of both clusters, the header has to deal with the received messages from both clusters, which is a hard work for the header. So, in both cases, we adopt some methods to let one of the clusters wait until the other completes most of its work. As explained above, two clusters with overlap area are separated in term of time, which means there is only one signal that can be sent out each time. When there is no overlap area and all clusters touch each other (see Figure 8), the number of signals will reach to its upper bound. The upper bound is easy to be worked out as follows:

\[
N_{\text{signal}} = \frac{S_{\text{whole}}}{\pi R^2 \left[ (2\sqrt{3} - \pi) / \pi \right]}, \tag{21}
\]

where \( S_{\text{whole}} \) is the whole event area with distributed sensors and the denominator is the actual area occupied by each clusters.

Using the method above, Bee-Sensor-C can take full advantage of cluster and significantly reduce routing overhead. In the next section, we will compare the performance of Bee-Sensor-C with the existing protocols through simulations.

5. Performance Evaluation

The simulations were conducted in two scenarios. We present the performance of Bee-Sensor-C against three existing protocols: BeeSensor, IEEABR, and FF-Ant in the first scenario where there is only an event that occurred in the network. Then, in the second scenario, we evaluate the performance of Bee-Sensor-C when there are two or more events detected in the network.

5.1. Simulation Environment. The program used in our simulations is written in JAVA and implements the network framework specified by ISO. The lower layer models the IEEE 802.11 network closely and carefully. The IP layer implements these routing protocols taken into comparison. In this section, we will comprehensively evaluate the integrated performance of Bee-Sensor-C through numerous simulations in a random deployed network running event-driven applications.

There are two static scenarios in the simulations. In the first scenario, the sensor nodes are randomly distributed in a square of 200 m \( \times \) 200 m and the node density increases as the number of nodes increases. Moreover, there is only one event that occurred and the nodes near the event will act as the source nodes. Besides, we evaluate Bee-Sensor-C in the form of scalability in the second scenario where the average node density remains the same and there are two events that occurred in the network.

In both scenarios, the distributed coordination function (DCF) of IEEE 802.11 is used to implement the MAC layer. The radio propagation range for each node is 40 m. The basic data rate is 1 Mbps and PLCP data rate is 2 Mbps. In addition, we consider the nodes in the event-sub-network as data sources using the CBR traffic model with data rate of 2 kbps for duration of 10 s. The parameters (\( \alpha, \beta, \omega_1, \omega_2, \eta, \) and \( \lambda \)) are set to (40, 10, 0.3, 0.4, 50, and 0.1), which are the result of our mathematical analysis and repeated simulations. Furthermore, the set of results recorded are average over ten independent simulation results. Other simulation settings in both scenarios are listed in Table 1, respectively. In Table 1, the size of area is equal to \( X_i \times Y_i \), where \( i \) represents five different areas and \( i \in \{1, 2, 3, 4, 5\} \).

5.2. Performance Evaluation Metrics. From the results obtained from our simulation experiments, we define the following performance metrics that we use to compare the performance of different protocols.

Packet delivery rate: it is the ratio of total number of data received at the sink node to the total number of data
Table 1: Simulation parameters.

| Parameter                        | Values in scenario 1 | Values in scenario 2 |
|----------------------------------|----------------------|----------------------|
| Data packet size (bytes)         | 256                  | 256                  |
| Control packet size (bytes)      | 24                   | 24                   |
| Initial energy (J)               | 3                    | 3                    |
| The number of nodes              | 100, 200, 300, 400, 500 | 14 \times 14, 18 \times 18, 22 \times 22, 26 \times 26, 30 \times 30 |
| The size of area (m²)            | 200 \times 200       | 196 \times 196, 252 \times 252, 308 \times 308, 364 \times 364, 420 \times 420 |
| The coordinate of event sources  | (50, 50)             | (30, \frac{Y_i}{2}), (110 + (i - 1) \times 28, \frac{Y_i}{2}) |
| The time when the event occurred (s) | 10               | 110, 10              |
| The radius of event-sub-network (m) | 20                | 20                  |
| The coordinate of sink           | (200, 100)           | (X_i, \frac{Y_i}{2}) |
| Convergence degree               | 1/3                  | 1/3                  |
| $E_c$ (nJ/bit)                   | 5                    | 5                    |
| Simulation time                  | 200 s                | 300 s                |


generated by all source nodes in the network. In Bee-Sensor-C, the data received at the sink node have been aggregated by CH.

Control overhead: it is the total size of control packets needed by a routing protocol for one kbits data packets received successfully at the sink (bits/kbits). Energy efficiency: it is the energy consumed in delivering one kbits data packets to sink node (J/kbits).

Energy standard deviation: it is defined as the average variation for energy of all nodes in the network with unit joules (J). An energy-aware routing protocol should not only aim for energy efficiency but also for balancing the network energy consumption.

Energy consumption: it is the total energy consumed by the nodes in the network during the period of experiment with unit Joules.

Routing building time: it is the average time building the multipath takes. In Bee-Sensor-C, it should include the time that constructing cluster takes but exclude the data transmission time in cluster.

Latency: it is defined as the difference in time when an event packet is generated at a source node and it is eventually received at the sink node. In Bee-Sensor-C, the source nodes should include all the nodes in the event-sub-network apart from CH.

5.3. Discussion on Simulation Results

5.3.1. Comparison of Bee-Sensor-C, BeeSensor, IEEABR, and FF-Ant in Scenario 1. In this set of experiments, we compare Bee-Sensor-C with BeeSensor, IEEABR, and FF-Ant in a static scenario. The results of our experiments are shown in Figures 9–14.

(1) Control Overhead. Figure 9 shows the control overhead of all protocols. Bee-Sensor-C has the smallest value among them and remains consistently unchanged. This is mainly due to the technique of clustering and restricted flooding in the phase of multipath construction. The dynamic clustering method in Bee-Sensor-C effectively reduces the number of
parallel event sources, which contributes to the high packet delivery rate because of the less traffic load. However, in BeeSensor, each event node needs to broadcast the scout agents to construct respective multipath, which directly leads to more collisions and rapidly increased control overhead with the increasing number of source nodes. Therefore, the performance of BeeSensor is poor, especially with the increase of node density. For the same reason, FF-Ant periodically floods the parallel forward ants causing large control traffic and congestion. This also leads to its lowest packet delivery rate and most control overhead. The control overhead of IEEABR is also higher due to its proactive nature and parallel control traffic transmission.

(2) Energy Efficiency. Figure 10 describes the energy efficiency of the protocols. Bee-Sensor-C has the best performance in terms of energy efficiency in this scenario. As the number of nodes increases, more traffic is generated in the network, and the total energy consumption rises unavoidably. Due to smoothly increased control traffic and high packet delivery rate, Bee-Sensor-C’s performance is the best. BeeSensor has large control traffic that ultimately leads to high energy consumption. For the same reason, IEEABR also has higher total energy consumption. FF-Ant has much longer route discovery process and the lowest packet delivery rate, which directly results in its worst performance in energy efficiency.

(3) Energy Standard Deviation. In Figure 11, it can be observed that Bee-Sensor-C presents a relatively stable variance and the lowest value of energy standard deviation. An algorithm should try to maximize average remaining energy of nodes together with a small standard deviation in order to prolong...
the network lifetime. The lowest standard deviation attained by Bee-Sensor-C is due to the fact that the design of Bee-Sensor-C takes the minimum energy nodes, average residual energy, and energy deviation into consideration and gets rid of abnormal node in the phase of multipath construction. Moreover, the enhanced multipath method can provide more paths, which is beneficial to utilize the energy of all nodes at a comparable rate. BeeSensor and FF-Ant have larger energy deviation due to their relatively more energy consumption and no concern about average residual energy and energy deviation of paths. However, IEEABR has the worst performance in standard deviation because of its unicast ant-forwarding policy and more energy consumption.

(4) Delivery Rate. Figure 12 shows the delivery rate of the protocols under evaluation. It is obvious that the delivery rate of Bee-Sensor-C remains close to the maximum value with the increased number of nodes. Higher control traffic and data traffic cause more collisions and packet loss. Although the node density is increasing, the cluster structure in Bee-Sensor-C significantly reduces the network traffic so that it can achieve the best performance in delivery rate. However, due to more collisions and congestion, the other protocols have worse delivery rate with the increment of node density.

(5) Latency. The latency of protocols is plotted in Figure 13. Bee-Sensor-C has better performance compared with other protocols mainly due to its fewer collisions. In Bee-Sensor-C, the paths between CH and sink are robust. Although constructing cluster and collecting the data in the cluster take some time in Bee-Sensor-C, the time is relatively short against delay brought about by frequent collisions of data and control packets in other protocols. Moreover, with the increase of node density, the collisions will be more and more serious, which results in more events waiting. These reasons are supported by the fact that the result of Bee-Sensor-C is the best in this scenario.

(6) Routing Building Time. The routing building time of the protocols is shown in Figure 14. Parallel data and control packet transmission cause more and more collisions with the increase of node density, which makes a protocol fail to discover a path. This in turn brings in further congestion. However, in Bee-Sensor-C, there is almost no collision because only CH needs building the path to sink. This explains the shortest routing building time of Bee-Sensor-C. Due to flooding policy, the routing building time of BeeSensor is much longer than that of IEEABR. And FF-Ant has the worst performance due to the most frequent collisions when building the route.

5.3.2. Evaluation of Bee-Sensor-C in Scenario 2. The main purpose of this experiment is to evaluate the performance of Bee-Sensor-C when there are two or more events detected in the network, which can reflect the scalability and robustness of Bee-Sensor-C. In the experiments, we call the protocol that adopts “cluster reuse” as Bee-Sensor-C. Otherwise, the protocol is called as “Bee-Sensor-C without CR.” Because Bee-Sensor-C has better performance than other protocols above in similar scenarios, we only compare Bee-Sensor-C to Bee-Sensor-C without CR. The results of our experiments are shown in Figures 15–17.

(1) Control Overhead and Energy Efficiency. Figure 15 plots the control overhead versus the number of nodes, showing the superiority of Bee-Sensor-C over Bee-Sensor-C without CR in restricting the number of control packets. It shows that 10% of overhead in Bee-Sensor-C is reduced. Due to making most use of living cluster and restricting the number of nodes participating in flooding forward scout, Bee-Sensor-C can appropriately reduce the number of network control packets with the network scale increasing. Figure 16 plots the energy efficiency versus the network size. It shows that the energy consumption in Bee-Sensor-C is less than that in Bee-Sensor-C without CR, especially when there are more and more
nodes. It attributes to the fact that Bee-Sensor-C introduces less overhead.

(2) Routing Building Time. Figure 17 shows that Bee-Sensor-C needs shorter routing building time than Bee-Sensor-C without CR. Beside the fewer collisions due to reducing control traffic, the mechanism also helps in that the intermediate CH can directly reply with backward scouts. These reasons are supported by the result in Figure 17 in this scenario. Although cluster maintenance should take some energy in Bee-Sensor-C, it is worthy to consider the benefits of less routing overhead and routing building time. Moreover, Bee-Sensor-C has better scalability which is a challenging issue in the design of routing protocols for WSNs.

6. Conclusion

In this paper, we proposed an energy-efficient and scalable multipath routing protocol based on dynamic clustering and foraging behavior of bee swarm for WSNs, that is, Bee-Sensor-C. By carrying out dynamic clustering and multipath method, Bee-Sensor-C improves the whole network performance. Via simulations, Bee-Sensor-C outperforms other protocols in metrics of control overhead, energy consumption, packet delivery rate, energy standard deviation, and latency. For instance, in the first scenario, control overhead of Bee-Sensor-C approximately reduces by 40.8%, 53%, and 66% as compared with that of Bee-Sensor, IEEABR, and FF-Ant, respectively. And in terms of the packet delivery rate, Bee-Sensor-C achieves an improvement of about 18.7% over BeeSensor, 16% over IEEABR, and 30.9% over FF-Ant. Moreover, Bee-Sensor-C well supports multicluster scenario. In the second scenario, Bee-Sensor-C reduces 10% control overhead of Bee-Sensor-C without using “cluster reuse” method. In the future, we intend to improve our algorithm to make it compatible with more application scenes and different sensor networks, such as dynamic scenario. And then we will implement the proposed algorithm on a real WSN hardware.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work was supported by the National Natural Science Foundation of China under Grants no. 61271176 and no. 61401334, the National Science and Technology Major Project under Grants no. 2011ZX03001-007-01 and no. 2013ZX03004007-003, the Fundamental Research Funds for the Central Universities, and the 111 Project (B08038).

References

[1] M. Leinonen, M. Codreanu, and M. Juntti, “Distributed joint resource and routing optimization in wireless sensor networks via alternating direction method of multipliers,” IEEE Transactions on Wireless Communications, pp. 5454–5467, 2013.

[2] Z. Xiong, A. D. Liveris, and S. Cheng, “Distributed source coding for sensor networks,” IEEE Signal Processing Magazine, vol. 21, no. 5, pp. 80–94, 2004.

[3] J. Evans, D. Raychaudhuri, and S. Paul, “Overview of wireless, mobile and sensor networks in GENI,” GENI Design Document, Wireless Working Group, 2006, http://groups.geni.net/geni/wiki/OldGPDesignDocuments.

[4] F. Ren, J. Zhang, T. He, C. Lin, and S. K. D. Ren, “EBRP: energy-balanced routing protocol for data gathering in wireless sensor networks,” IEEE Transactions on Parallel and Distributed Systems, vol. 22, no. 12, pp. 2018–2125, 2011.

[5] K. Lee and H. Lee, “A self-organized and smart-adaptive clustering and routing approach for wireless sensor networks,” International Journal of Distributed Sensor Networks, vol. 2012, Article ID 156268, 13 pages, 2012.

[6] A. M. Zungeru, L.-M. Ang, and K. P. Seng, “Termitic-hill: performance optimized swarm intelligence based routing algorithm for wireless sensor networks,” Journal of Network and Computer Applications, vol. 35, no. 6, pp. 1901–1917, 2012.

[7] S. Ehsan and B. Hamdaoui, “A survey on energy-efficient routing techniques with QoS assurances for wireless multimedia sensor networks,” IEEE Communications Surveys & Tutorials, vol. 14, no. 2, pp. 265–278, 2012.

[8] J. Yang, M. Xu, W. Zhao, and B. Xu, “A multipath routing protocol based on clustering and ant colony optimization for wireless sensor networks,” Sensors, vol. 10, no. 5, pp. 4521–4540, 2010.

[9] N. A. Pantazis, S. A. Nikolidakis, and D. D. Vergados, “Energy-efficient routing protocols in wireless sensor networks: a survey,” IEEE Communications Surveys & Tutorials, vol. 15, no. 2, pp. 551–591, 2013.

[10] N. A. Alrajeh, M. S. Alabed, and M. S. Elwahiby, “Secure ant-based routing protocol for wireless sensor network,” International Journal of Distributed Sensor Networks, vol. 2013, Article ID 326295, 9 pages, 2013.

[11] M. Saleem, G. A. Di Caro, and M. Farooq, “Swarm intelligence based routing protocol for wireless sensor networks: survey
and future directions,” *Information Sciences*, vol. 181, no. 20, pp. 4597–4624, 2011.

[12] F. Çelik, A. Zengin, and S. Tuncel, “A survey on swarm intelligence based routing protocols in wireless sensor networks,” *International Journal of Physical Sciences*, vol. 5, no. 14, pp. 2118–2126, 2010.

[13] M. Saleem, I. Ullah, and M. Farooq, “BeeSensor: an energy-efficient and scalable routing protocol for wireless sensor networks,” *Information Sciences*, vol. 200, pp. 38–56, 2012.

[14] A. Da Silva Rego, J. Celestino, A. Dos Santos, E. C. Cerqueira, A. Patel, and M. Taghavi, “BEE-C: a bio-inspired energy efficient cluster-based algorithm for data continuous dissemination in Wireless Sensor Networks,” in *Proceedings of the 18th IEEE International Conference on Networks (ICON ’12)*, pp. 405–410, December 2012.

[15] D. Karaboga, S. Ökdem, and C. Ozturk, “Cluster based wireless sensor network routing using artificial bee colony algorithm,” *Wireless Networks*, vol. 18, no. 7, pp. 847–860, 2012.

[16] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, “Energy-efficient communication protocol for wireless microsensor networks,” in *Proceedings of the 33rd Hawaii International Conference on System Sciences*, pp. 3005–3014, IEEE Computer Society, Washington, DC, USA, January 2000.

[17] S. Yi, J. Heo, Y. Cho, and J. Hong, “PEACH: power-efficient and adaptive clustering hierarchy protocol for wireless sensor networks,” *Computer Communications*, vol. 30, no. 14–15, pp. 2842–2852, 2007.

[18] Y. M. Lu and V. W. S. Wong, “An energy-efficient multipath routing protocol for wireless sensor networks,” *International Journal of Communication Systems*, vol. 20, no. 7, pp. 747–766, 2007.

[19] M. Kim, E. Jeong, Y.-C. Bang, S. Hwang, and B. Kim, “Multipath energy-aware routing protocol in wireless sensor networks,” in *Proceedings of the 5th International Conference on Networked Sensing Systems (INSS ’08)*, pp. 127–130, Kanazawa, Japan, June 2008.

[20] A. Pandya and M. Mehta, “A novel energy efficient routing approach using multipath ring routing and clustering for WSN,” in *Proceedings of the International Information Technology Conference (CUBE ’12)*, pp. 138–143, September 2012.

[21] Y. Zhang, L. D. Kuhn, and M. P. J. Fromherz, “Improvements on ant routing for sensor networks,” in *Proceedings of the 4th International Workshop on Ant Colony Optimization and Swarm Intelligence (ANTS ’04)*, pp. 154–165, 2004.

[22] T. Camilo, C. Carreto, J. S. Silva, and F. Boavida, “An energy-efficient ant-based routing algorithm for wireless sensor networks,” in *Ant Colony Optimization and Swarm Intelligence*, pp. 49–59, Springer, Berlin, Germany, 2006.

[23] A. M. Zungeru, K. P. Seng, L.-M. Ang, and W. Chong Chia, “Energy efficiency performance improvements for ant-based routing algorithm in wireless sensor networks,” *Journal of Sensors*, vol. 2013, Article ID 759654, 17 pages, 2013.

[24] H. F. Wedde, M. Farooq, T. Pannenbaecker et al., “BeeAdHoc: an energy efficient routing algorithm for mobile ad hoc networks inspired by bee behavior,” in *Proceeding of the Genetic and Evolutionary Computation Conference (GECCO ’05)*, pp. 153–160, New York, NY, USA, June 2005.

[25] M. Saleem and M. Farooq, “BeeSensor: a bee-inspired power aware routing protocol for wireless sensor networks,” *Applications of Evolutionary Computing*, pp. 81–90, 2007.

[26] D. Karaboga, B. Gorkemli, C. Ozturk, and N. Karaboga, “A comprehensive survey: artificial bee colony (ABC) algorithm and applications,” *Artificial Intelligence Review*, vol. 42, no. 1, pp. 21–57, 2014.

[27] C. Bettstetter, “On the minimum node degree and connectivity of a wireless multihop network,” in *Proceedings of the 3rd ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC ’02)*, pp. 80–91, June 2002.

[28] M. Saleem, S. A. Khayam, and M. Farooq, “A formal performance modeling framework for bio-inspired Ad Hoc routing protocols,” in *Proceeding of the 10th Annual Genetic and Evolutionary Computation Conference (GECCO ’08)*, pp. 103–110, New York, NY, USA, July 2008.