A Collaborative Self-Governing Privacy-Preserving Wireless Sensor Network Architecture Based on Location Optimization for Dynamic Service Discovery in MANET Environment

Cong Gao, Jianfeng Ma, and Shangwei Zhang

School of Computer Science and Technology, Xidian University, Xi'an 710071, China

Correspondence should be addressed to Cong Gao; glidergao@gmail.com

Received 2 July 2014; Revised 23 August 2014; Accepted 9 September 2014

Academic Editor: Rongbo Zhu

Due to the characteristics of a MANET, none of the existing solutions for service discovery work well in decentralized mobile environments. In this paper, we propose a collaborative self-governing privacy-preserving wireless sensor network architecture to address the issue of service discovery in MANET environment. The proposed architecture is able to dynamically adjust the working modes between directory-based and directory-less modes according to the network status of a MANET. The dynamic network status of a MANET is monitored by a wireless sensor network. Two location optimization algorithms are developed for topology control and the preservation of user location privacy. Simulation results show that the proposed architecture can greatly improve the performance of service discovery.

1. Introduction

With the booming development of the Internet, more and more services are deployed on the web. Hence, the need for service discovery mechanisms is becoming increasingly urgent. Much research attention has been drawn to the designing of service discovery mechanisms. Meditskos and Bassiliades [1] developed a web service discovery framework using OWL-S advertisements. Junghans et al. [2] provided a review of formalisms for semantic web service discovery and developed distinct formalisms to describe functionalities and service requests. The proposal presented by Santhanam and Balasundaram [3] employed $k$-means clustering to address the problem of web service discovery. In addition, service discovery methods which deal with functional and nonfunctional properties of services were also extensively studied [4]. Some techniques for web service discovery were contributed in the form of a patent [5]. In general, the service discovery could be conducted in three styles: reactive way, proactive way, and hybrid way. In reactive way, a user node which has a need of some kind of service issues a service query. Other nodes which received the service query may answer it by issuing a service reply. The nodes which answer the service query could be a directory node, service node, or user node. In proactive way, service information is periodically advertised by service nodes. Both directory nodes and user nodes will receive the service advertisements. If a user node which has a potential need of service happens to receive an appropriate service advertisement could directly use the service information, rather than issuing a service query to search service information.

Generally, the reactive way only generates network traffic when there is a need. To some degree, this leads to as less message overhead as possible. During the query process, the original service query might be forwarded by a node which is unable to answer it. In order to avoid flooding the whole network, it is necessary that a subtle time-to-live (TTL) value should be set. Here, the dilemma is that a small TTL value will limit the transmission range and then brings bad influence in the service discovery, while a large TTL value will expand the transmission range and then floods the query to a wide area. Since service nodes do not actively issue service advertisements in reactive way, both the latency and the failure rate of the service discovery are high. Similarly, the TTL value of periodic service advertisements should be cautiously selected in proactive way. Moreover, in
order to mitigate the flooding effect caused by periodic service advertisements, the frequency of service advertisement should be elaborately tuned. As service nodes periodically issue service advertisements no matter whether there is a need for service information, both the latency and the failure rate of the service discovery in proactive way are lower than that in reactive way. However, this in turn constantly imposes a certain amount of message overhead on the whole network. As both the reactive way and the proactive way have advantages and disadvantages, the hybrid way combines the two complementary ways. However, since disadvantages of both ways are introduced, how to achieve a tacit cooperation of the two ways is challenging.

In this paper, we propose a collaborative self-governing privacy-preserving wireless sensor network (WSN) architecture for dynamic service discovery in MANET environment. Our model is based on the famous Chord protocol and could operate in both directory-based mode and directory-less mode. The WSN which consists of directory nodes greatly facilitates the service discovery. The topology control for the WSN is conducted through local location optimization and global location optimization. The location privacy of a user is preserved based on a direction-probing algorithm used in the local location optimization. The key function of our model is the autonomic mode switch between the directory-based mode and the directory-less mode.

The rest of this paper is organized as follows. Section 2 provides the related works about service discovery architectures and discusses the existing problems. Section 3 elaborates our collaborative self-governing privacy-preserving service discovery architecture. In Section 4, we evaluate the proposed model by simulation and present a detailed analysis of the simulation results. Section 5 concludes the paper and highlights the future work.

2. Review of Service Discovery Architectures

Traditionally, service discovery architectures are classified into two types: directory-based architecture and directory-less architecture.

2.1. Directory-Based Architecture. In directory-based architecture, a directory structure could be implemented either in centralized manner or in distributed manner.

For a centralized directory, all service information is stored in the same place. All services in the whole network should be registered with a centralized repository. At the same time, the centralized repository is responsible for all service queries. Examples of centralized directory are as follows: the service location protocol (SLP) [6] suggested by IETF; the jini technology [7] developed by Sun Microsystems, the salutation protocol [8] proposed by IBM, the universal plug and play device architecture (UPnP) [9] specified by Microsoft, the Bluetooth service discovery protocol [10] presented by Bluetooth consortium, the secure wide-area service discovery service (SDS) [11], and the intentional naming system (INS) [12]. Generally speaking, a centralized manner means attractive simplicity, inevitable single point of failure, poor scalability, and weak mobility. On the contrary, a distributed manner indicates certain complexity, robustness, good scalability, and strong mobility. Taking the characteristics of a MANET environment into consideration, we hold that the distributed manner is preferable. Hence, we omit the details of the above centralized directories and focus on the instances of distributed directory.

For a distributed directory, the problem of wide-area service discovery needs to be focused on. There are two aspects to the concept of wide-area service discovery: one is the replication of service registration, and the second is the forwarding of service query. Since directory agents reside in various locations, the service registrations they received are usually different. Then, each directory agent only knows about a portion of service information in the whole network. If there is no replication scheme in a distributed directory, service information is only locally available around the particular directory agent with which it was registered. Rather, a distributed directory with a replication scheme enables the communication among directory agents. By replication of service information, each directory agent becomes aware of all services registered in the whole network. Suppose there is a distributed directory which does not support replication of service registration. For the purpose of improving the success rate of the service discovery, it is reasonable that a service query which could not be answered locally should be forwarded to another directory agent. This operation relies on a forwarding scheme provided by the distributed directory. Recently, a considerable amount of work has been done on the distributed directory. Based on the organization of the directory agents, distributed directory could be divided into four categories: backbone-based [13–15], cluster-based [16–19], DHT-based [20, 21], and other distributed directories [22, 23].

Kozat and Tassiulas [13] provided a virtual backbone which facilitates the registration and search of services. The virtual backbone is constructed by a subset of nodes in the network, and these nodes compose a dominating set. Though there is no replication scheme, nodes belong to the backbone forward service queries to each other when they cannot be answered locally. However, there is a flaw that the destination of the forwarding operation is random. The proposal [14] presented by Sailhan and Issarny remedied the flaw by forwarding an unsolved service query to the nodes which are expected to possess the corresponding service information. The selection of destination to which a service query is forwarded is based upon the exchange of profiles among directory agents. The protocol [15] proposed by Koubaa and Fleury aimed to build a backbone which could cover all service information in the whole network. Nevertheless, as only a service provider is allowed to join the backbone, it is likely that an integral backbone might be broken down into several isolated parts. In this case, the global communication of the backbone is cut off. Consequently, the disposal of service queries is crippled.

Klein et al. [16] presented a semantic overlay of hierarchical service rings. A service ring is composed of a cluster of service providers and is considered to be a directory, since the functions of accepting service registration and answering service query are contributed in the form of a service ring.
Although replication is not supported, a service ring forwards a service query which it is unable to answer. Klein and König-Ries [17] proposed that services are dynamically organized into multilayer clusters. The authors make an assumption that there exists a common ontology which could be used to describe the services within the whole network. Upon receiving a service query, a node checks its leaf-level cluster. If the service query cannot be answered, higher-level clusters are examined in the pursuit of a suitable service. It is worth mentioning that nodes are clustered based on the physical proximity and the semantic proximity [16, 17]. Schiele et al. [18] suggested that clusters are formed according to the similarity of mobility patterns. A clusterhead is always ready to answer service queries. The remaining nodes in a cluster go to sleep when idle. Tyan and Mahmoud [19] presented a service discovery approach which clusters the nodes according to physical proximity. A gateway of each cluster plays the role of a directory. Wide-area service discovery is feasible since an unsolved service query could be forwarded to another gateway.

As mentioned above, when a replication scheme is not present, an unsolved service query should be forwarded to other nodes. However, it is not easy to determine an appropriate destination for an unsolved service query. Unlike backbone-based and cluster-based models, an exciting feature that the DHT-based approaches inherently possess is hash techniques which could facilitate the provision of location information. Seada and Helmy [20] developed a rendezvous-based architecture. The basic idea is that the whole network is divided geographically and each geographical area is responsible for a portion of service information in the network. Service information is mapped to a geographical area according to a hash-table-like mapping scheme. A part of nodes within a geographical area are selected to maintain the mapped information. Since the hash function used by service query is the same as the hash function used in the mapping of service information, the proper forwarding destination of a service query could be readily obtained by a directory agent. Sivavakesar et al. [21] described a mechanism which is similar to the approach proposed in [20] except for the fact that the mapped information is handled by all nodes within a geographical area.

Other approaches which do not involve backbone, cluster, or DHT techniques are also notable. Lee et al. [22] presented a service discovery and delivery protocol called Konark. The proposed method is decentralized and targeted at a MANET environment. Under a pure peer-to-peer design, each node in the network has a database which stores the service information provided by other nodes. Jeong et al. [23] developed an ad hoc name service (ANS) system for IPv6 MANET to facilitate the service discovery and name-to-address resolution. It is assumed that each node could be configured with a site-local scoped IPv6 unicast address by ad hoc stateless address autoconfiguration [24, 25].

2.2. Directory-Less Architecture. As a MANET environment is born with no infrastructure, it is argued that a directory-less architecture is more appropriate than a directory-based architecture [26]. However, there is an important distinction between a directory-less architecture and a directory-based architecture: there are no directory agents in a directory-less architecture. Namely, there are only user agents and service agents in the whole network. The missing of the organization and maintenance for directory agents reduces the complexity of service discovery architecture: user agents issue service queries and service agents issue service advertisements. Though service queries are generated on demand, the number of replicates of a service query during the forwarding process could become very large. Furthermore, in order to achieve a decrease in latency and an increase in service availability, the number of service advertisements is also considerable. In this case, a pursuit of good service discovery performance might be prone to bring about a serious flooding problem.

Much research attention has been drawn to the operations of service queries and service advertisements. Chakraborty et al. [27] proposed a group-based service discovery (GSD) protocol for pervasive environment. In general, the protocol consists of two components. The first component is a P2P caching scheme dealing with the spread of service advertisements. A node stores the received service advertisements in a local database called service cache. A least-remaining-lifetime cache replacement policy is used to update the cache. The second component is a group-based intelligent forwarding strategy for regulating the forwarding of service queries. In specific, an unsolved service query is just forwarded to the nodes which possess matching service information. The determination of destination nodes is based on the information contained in the service cache. The reply info caching enhanced flexible forward probability (RICFPP) service discovery protocol presented by Gao et al. [28] also consists of two parts: reply info caching (RIC) and flexible forward probability (FFP). The RIC is a technique similar to the P2P caching scheme in [27]. In particular, a node could cache the service information in a received service reply. For a received service query, if there exists a match in the cache, the service query could be answered conveniently. The essence of the FFP is that the forward probability of a service query is in inverse proportion to the number of hops it has already traveled. Nedos et al. [29] introduced an optimal forwarding mechanism for service advertisements. By monitoring the neighbors, a node merely forwards the received service advertisements to a part of its one-hop neighbors. Therefore, the broadcasting of service advertisements is turned into several unicasts. Nidd [30] set the target application environment to single-hop short-range wireless systems. For each node, the broadcasting of service advertisements is scheduled to take place at regular intervals. The task of scheduling is performed by an exponential back-off algorithm. The dissemination strategies proposed by Campo et al. [31] and Lee et al. [32] are similar that they both cache service information contained in received service advertisements. Moreover, only service advertisements which are not expired and have not been seen recently are broadcasted.

2.3. Problems. As the directory-based architecture and the directory-less architecture are complementary, hybrid architecture is another option for service discovery in a MANET environment. For a node in hybrid architecture, if there
is no directory agent within its radio range, both service advertisements and service queries are sent in the same way of directory-less architecture. Otherwise, service information contained in service advertisements is registered with directory nodes. Service queries could be answered by both directory agents and service agents. Though a great deal of research has been done to the above three types of service discovery architectures, it is still controversial that which one is superior to the other two. It is recognized that the difficulty of comparison is largely due to the characteristics of a MANET environment. For instance, the nodes in a MANET might possess various degrees of mobility. Moreover, both the velocity and the moving direction of a node are unpredictable. The regular communication among nodes in the whole network should not be interrupted by the node mobility. Since a MANET is self-organized, there are no constraints on the joining and leaving of a node. The appearance of a new participator or departure of an existing participator could bring a significant change to the network topology. Again, the communication in the whole network should not be influenced.

Considering a MANET with directory-based architecture, the TTL value of service query should at least guarantee that a service query could arrive at a directory node. If the frequency of service query is low, the message overhead introduced by service registration and directory maintenance is likely to surpass the message overhead brought in by service advertisement with a directory-less architecture. For directory-less and hybrid architectures, large TTL values of both service query and service advertisement could improve the performance of service discovery. However, there is a compromise between flooding problem and performance enhancement. When there are a large number of service queries, a directory-less architecture has to increase the number of service advertisements. But since both service query and service advertisement are simply broadcasted, a risk of network congestion still could not guarantee a satisfactory performance of service discovery. Ververidis and Polyzos [33] pointed out that it could be useful to develop a flexible architecture which is able to adjust its parameters and working modes between directory-based and directory-less modes according to the status of a MANET.

3. A Collaborative Self-Governing Privacy-Preserving WSN Architecture for Dynamic Service Discovery

3.1. Network Model

3.1.1. Nodes. In our architecture, there are three possible roles for a mobile node: service provider, service requestor, and service directory. In order to perform the operations which facilitate the service discovery process, the above three types of nodes are equipped with three key components service agent (SA), user agent (UA), and directory agent (DA), respectively. For the sake of simplicity, in the remainder of this paper, we refer to service provider nodes, service requestor nodes, and service directory nodes as SAs, UAs, and DAs, respectively. To facilitate the presentation of our model, we consider a MANET consists of DAs, SAs, and UAs. In addition, there is a WSN composed of all DAs.

3.1.2. Directory Agent. Each DA is a wireless sensor node. DAs offer a place storing service information and SAs register service information with DAs. Then, DAs answer the service queries issued by UAs based on the service information they have. The existence of DA is to improve the effectiveness of the service discovery architecture. The WSN constituted by all the DAs is based on Chord [34]. Chord is a scalable distributed lookup protocol which has been applied to service discovery [35–37]. The most important operation that Chord provides is mapping a given key onto a node. By consistent hashing, each data item could be associated with a unique key. The \langle key, data item \rangle pair is stored at the corresponding node to which the key maps. The mechanisms facilitating the joining and leaving of node could guarantee the normal operations of the whole network. In classic sensor networks, all sensor nodes are scheduled to send the obtained information to a single sink node. Though this approach eases the routing work, authors of [38, 39] point out that the nodes which are in vicinity of the sink node bear heavy workload of relaying. As a result, the batteries of these nodes are drained out much earlier than other nodes. Thus, there is no sink in our WSN of all the DAs. The status of the whole network is summarized and presented in a monitoring token. In clustered sensor networks, the principle of clusterhead rotation is studied extensively in [40–42]. However, the election and rotation of clusterhead inevitably introduce extra overhead. On the contrary, since the consistent hashing mechanism employed by Chord maps the keys of data items uniformly, the querying process of service is inherently load-balanced for all the DAs. This feature also helps to avoid a quick energy exhaustion of certain DAs.

3.1.3. Unified Service Information Management. Our service discovery architecture employs a unified service information management scheme. It is necessary that all DAs, SAs, and UAs which participated in the architecture are aware of the scheme. In particular, there is a list of properties which is used for profiling a service. Each property has a number of values. We denote the list of properties and values by \( L \). The most important operation that Chord provides is mapping a given key onto a node. By consistent hashing, each data item could be associated with a unique key. The \langle key, data item \rangle pair is stored at the corresponding node to which the key maps. The mechanisms facilitating the joining and leaving of node could guarantee the normal operations of the whole network. In classic sensor networks, all sensor nodes are scheduled to send the obtained information to a single sink node. Though this approach eases the routing work, authors of [38, 39] point out that the nodes which are in vicinity of the sink node bear heavy workload of relaying. As a result, the batteries of these nodes are drained out much earlier than other nodes. Thus, there is no sink in our WSN of all the DAs. The status of the whole network is summarized and presented in a monitoring token. In clustered sensor networks, the principle of clusterhead rotation is studied extensively in [40–42]. However, the election and rotation of clusterhead inevitably introduce extra overhead. On the contrary, since the consistent hashing mechanism employed by Chord maps the keys of data items uniformly, the querying process of service is inherently load-balanced for all the DAs. This feature also helps to avoid a quick energy exhaustion of certain DAs.

Definition 1. By description of a service we mean the character string desp = \( p_{V_{m}} = V_{m}p_{V_{m}}V_{m}p_{V_{m}} \) which is formed by concatenating the values of \( np - i \) properties sequentially, where \( 0 < i < np \) and \( 1 < s, j, k \leq 256 \). When \( i = 0 \), we
3.2. Directory-Based Mode. In directory-based mode, all DAs in the MANET form a Chord network. All nodes in the MANET are mobile. SAs register their service information with DAs. A service query issued by a UA is handled by DAs.

3.2.1. Service Registration and Deregistration in WSN. For service registration, the service information submitted to a DA consists of three parts: description of the service, ID of the service provider (e.g., company name), and URL for accessing the service. Note that only a complete service description can be used for service registration. There are 32 bytes to accommodate the ID of service provider. Though the HTTP protocol does not stipulate the maximum length of a URL, we limit it to a maximum of 2048 bytes. The format of service registration is shown in Figure 2.

Since the WSN is based on Chord, we denote the identifier space of the Chord network by $2^m$. In order to accommodate the total $\prod_{i=1}^{np} |p_i|$ different complete descriptions for profiling services, the inequality $2^m \geq \prod_{i=1}^{np} |p_i|$ should be satisfied.

In our model, we use the SHA-1 [43] as the consistent hash function to generate keys for both DAs and services. We denote the hash function by $\text{sHash}(\text{string})$. Upon receiving a valid service registration, a DA first sends a service registration acknowledgment to the SA which issued the service registration. Then, the DA extracts the description and the ID parts as a character string $\text{str}$. The key of the service is obtained by $\text{key} = \text{sHash}($str$)$. The $\langle \text{key}, \text{URL} \rangle$ pair is then sent to the DA which is responsible for the key. An SA could register its service information with an arbitrary DA within its radio range. As long as an SA receives a service registration acknowledgment, there is no need to register the service again. If the corresponding DA becomes out of the radio range when an SA is registering a series of services, the SA could contact another DA to register the remaining services. The service deregistration is conducted in a similar way to the service registration.

3.2.2. Key Generation and Searching in WSN. In unstructured P2P network, the searching based on a keyword is feasible. However, the consistent hashing of filename in structured DHT prohibits such operation in Chord. As a result, only the approach of exact match works. The inverted index proposed in [44] employs the mapping of $(\text{keyword}, \text{node})$. Then, the searching based on a keyword is implemented by redundancy storage. The major drawback of this approach is the problem of common keywords. The nodes which are responsible for common keywords suffer from heavy burden. In our model, a key is generated by hashing the character string of the description of a service concatenated by the service provider ID. The inclusion of the service provider ID resolves the above problem of common keywords.

When there is a potential need of certain service, several interested values are selected from the list $\ell$. By a UA, a service query with the same format as a service description is sent to a DA. Upon receiving a service query, a DA first checks whether there are missing values for properties. For a complete description, the DA which received the service query simply appends the IDs of service providers known to it and hashes the character string composed of the description and the ID. Then, the DA searches the obtained keys. However, it is very likely that values of some properties are missing due to uncertainty. Consequently, we have an incomplete description. For $i$ missing values, each one is filled by several most popular values queried in the past. We denote the number of most popular values by $mmpv$.

After the filling operation of the original incomplete description, we have $mmpv$ complete descriptions. Then, the IDs of service providers known to the DA are appended to the complete descriptions. We denote the number of service providers a DA is aware of by $nsp$. For a scenario of $i = 1$, $mmpv = 3$, and $nsp = 2$, the filling and appending process of an incomplete description is illustrated in Figure 3. Since there are six character strings of the descriptions concatenated by the service provider IDs, we could obtain six different keys for searching. The missing of property values is allowed, as it provides an open search result for a UA. However, since the keys are generated based on service information which is wishfully created by a UA and a DA, it is worth noting that both a complete description and an incomplete description are likely to yield no search results. For an existing service with a search key, the responsible DA will send a service reply to the original DA which issued the search. Then, the search result is fed back to the UA by the original DA.

3.2.3. Topology Control and Function Tuning for WSN. Since DAs are mobile in directory-based mode, the mobility has a strong impact on the topology of the WSN. Moreover, the unpredictable node failures also exert influence on the topology of the WSN. In our model, the topology control for WSN consists of two parts: local location optimization and global location optimization.
The local location optimization (LLO) is performed between a DA and the UAs which are communicating with the DA. For an individual DA, we divide its 360-degree radio range into six equal adjacent areas: \( a_i \) \((i = 1, 2, \ldots, 6)\). As shown in Figure 4, the upward direction represents north.

The two salient features of the local location optimization are privacy preservation and energy conservation. As a long communication distance requires more energy for radio transmission, it is beneficial that the communication distance between a DA and a UA could be shortened once they get in touch with each other. In our model, the result of a service query is sent to a UA by the original DA it queried. This procedure makes the shortening of communication distance even more important. Since the mobile nodes are equipped with omni-directional antennas, a DA has no idea of where a UA is and therefore is unable to determine a correct moving direction. For the sake of privacy protection, a UA is unwilling to share its location in any form (e.g., GPS coordinates). To address this problem, we develop a direction-probing algorithm to determine a moving direction for a DA. As shown in Figure 5, this algorithm contains a four-step movement: \( s_i \) \((i = 1, 2, \ldots, 4)\).

![Figure 4: Six areas.](image)

![Figure 5: Four-step movement.](image)

Considering a DA whose original location is \((x_0, y_0)\) and a UA whose location is unknown, the DA performs the following movements consecutively: \( s_1(N, \Delta y)\), \( s_2(E, \Delta x)\), \( s_3(S, \Delta y)\), and \( s_4(W, \Delta x)\). The values of \( \Delta x \) and \( \Delta y \) are both positive. After a four-step movement, the DA returns to its original location \((x_0, y_0)\). During each step, the variation of signal intensity is recorded. We denote an increase and a decrease of signal intensity by + and −, respectively. Then, the moving direction of the DA is determined according to Table 1.

When there are several UAs which are communicating with the DA, the desired moving directions related to different UAs might be various. Generally, it is considered that the energy consumption is in direct proportion to the communication distance. Hence, in order to reduce the total energy consumption, the sum of distances between the DA and each UA should be as small as possible. Considering two UAs \( u_1 \) and \( u_2 \), the communication distances between the two UAs and the DA are denoted by \( r_1 \) and \( r_2 \), respectively. In addition, we denote the signal intensities of the two UAs by \( s_1 \) and \( s_2 \), respectively. We make the assumption that the signal intensity and the communication distance follow the inverse square law \( s_i = \frac{\delta}{r_i^2} \), where \( \delta \) is a constant coefficient and \( i = 1, 2 \). The condition of equality for \( r_1 + r_2 = 2\sqrt{r_1 \cdot r_2} \) is \( r_1 = r_2 \). Moreover, \( r_1 = r_2 \) is equivalent to \( s_1 = s_2 \). Similarly, for \( n \) UAs, we have the condition of equality for the inequality \( r_1 + r_2 + \cdots + r_n \geq n\sqrt{r_1 \cdot r_2 \cdots \cdot r_n} \) as \( s_1 = s_2 = \cdots = s_n \). Thus, the moving direction of the DA is adjusted based on the idea of equalizing the signal intensities of all UAs. However, when there are several UAs, this equalization process could be complicated. Alternatively, we choose to synthesize the \( n \) moving directions determined for the \( n \) UAs with weight values. The weight values of the UAs are calculated in reverse...
Direction-Probing \((da, ua[\text{n}])\)

1. original signal intensity \(ua[j].s(0)\)
2. for \(i \leftarrow 1\) to 4
3. for \(j \leftarrow 1\) to \(n\)
4. if \(ua[j].s(i) > ua[j].s(i-1)\) then
5. signal variation \(ua[j].v(i) \leftarrow +\)
6. else signal variation \(ua[j].v(i) \leftarrow -\)
7. end if
8. \(s[j][i] \leftarrow ua[j].v(i)\)
9. end for
10. end for
11. for \(j \leftarrow 1\) to \(n\)
12. switch \(s[j]\)
13. case "+--": \(ua[j].d = a_1\), break
14. case "+-+": \(ua[j].d = a_2\), break
15. case "-++": \(ua[j].d = a_3\), break
16. case "--+": \(ua[j].d = a_4\), break
17. case "+-": \(ua[j].d = a_6\), break
18. case "-+-": \(ua[j].d = a_6\), break
19. end switch
20. end for
21. \(ua'[\text{n}] = \text{order (ascending, } ua[\text{n}], ua[\text{n}].s(0))\)
22. \(s.sum \leftarrow 0\)
23. for \(j \leftarrow 1\) to \(n\)
24. \(s.sum \leftarrow s.sum + ua'[j].s(0)\)
25. end for
26. for \(j \leftarrow 1\) to \(n\)
27. \(ua'[j].w \leftarrow s.sum/ua'[j].s(0)\)
28. end for
29. \(da.d = \text{synthesis}(ua'[j].d, ua'[j].w)\)

Algorithm 1: Direction-probing procedure.

The energy statuses and descriptions are given in Table 2.

| Status | Description |
|--------|-------------|
| \(S_n\) | Normal status. The remaining energy is \(E(da) > e_a\) |
| \(S_l\) | Low status. The remaining energy is \(e_b < E(da) \leq e_a\) |
| \(S_a\) | Alert status. The remaining energy is \(e_c < E(da) \leq e_b\) |
| \(S_s\) | Serious status. The remaining energy is \(E(da) \leq e_y\) |

by \(da.successor\). When \(da.predecessor\) or \(da.successor\) keeps decreasing, \(da\) becomes aware of the fact that its predecessor/successor is moving away from it. Once \(da.predecessor\) or \(da.successor\) is lower than a threshold \(s_{low}\), \(da\) initiates a direction-probing procedure to determine the direction of the predecessor/successor. Meanwhile, it sends a SLOW_DOWN message to the predecessor/successor for notification. Once a moving direction is determined, \(da\) starts to move toward the predecessor/successor in order to prevent the signal intensity from getting too low. In addition, the predecessor/successor which receives the SLOW_DOWN message will adjust its movement for the purpose of getting close to \(da\). Once the signal intensity between \(da\) and the predecessor/successor reaches a certain level, the predecessor/successor stops getting close to \(da\). We denote the signal intensity of this level by \(s_{normal}\). Note that when both \(da.predecessor\) and \(da.successor\) are greater than or equal to \(s_{normal}\), the \(da\) stops moving. For both \(da.predecessor\) and \(da.successor\), a compromise of directions might be needed. In this case, a composite moving direction is used by \(da\). The detailed connectivity maintenance algorithm is illustrated in Algorithm 2. Since the energy of a DA is limited, we hold that the function of connectivity maintenance stops when the remaining energy of a DA is critically low, and this feature is indicated by the last statement of Algorithm 2. A complete function tuning algorithm is described later.

As energy conservation is a critical issue in WSNs for DAs and network life, we develop a function tuning algorithm based on the remaining energy of a DA. Moreover, this algorithm also assists the topology maintenance. We denote the remaining energy of \(da\) by \(E(da)\). In specific, the remaining energy of a DA is classified into four statuses according to three critical values: \(e_a, e_b, \text{ and } e_y\). The four energy statuses \(S_n, S_l, S_a, \text{ and } S_s\) are detailed in Table 2.

When \(da\) is in the normal status, it is fully functional. Namely, it accepts service registrations, handles service queries, sends service replies, performs local location optimization and global location optimization, and so forth. Once \(da\) enters the low status, it stops performing local location optimization. When it turns into the alert status, an ALERT message is sent to both \(da.predecessor\) and \(da.successor\). As soon as \(da.predecessor\) and \(da.successor\) receive the ALERT message, they both stop the local location optimization temporarily (even when they are in the normal status). Now, the three DAs \(da, da.predecessor, \text{ and } da.successor\) are in preparation for the future serious status of \(da\). Once \(da\) enters the serious status, it disables all functions except for performing a data transfer. By the Chord protocol, a node which is about to leave the network should transfer all the \(\langle key, dataitem\rangle\) pairs it is responsible for to its neighbors.
ConnectivityMaintenance(da, da.predecessor, da.successor)
(1) repeat
(2) FLAG₁ ← false
(3) FLAG₂ ← false
(4) if da.predecessor.s < s_low then
(5)  FLAG₁ ← true
(6)  da.send(SLOW_DOWN, da.predecessor)
(7)  da.d₁ = da.direction-probing(da, da.predecessor)
(8)  if da.predecessor received SLOW_DOWN then
(9)   da.predecessor.d = da.predecessor.direction-probing(da.predecessor, da)
(10)  da.predecessor.move(da.predecessor.d)
(11) end if
(12) end if
(13) if da.successor.s < s_low then
(14)  FLAG₂ ← true
(15)  da.send(SLOW_DOWN, da.successor)
(16)  da.d₂ = da.direction-probing(da, da.successor)
(17)  if da.successor received SLOW_DOWN then
(18)   da.successor.d = da.successor.direction-probing(da.successor, da)
(19)  da.successor.move(da.successor.d)
(20) end if
(21) end if
(22) if FLAG₁ && FLAG₂ == true then
(23)  da.predecessor.w = (da.predecessor.s + da.successor.s)/da.predecessor.s
(24)  da.successor.w = (da.predecessor.s + da.successor.s)/da.successor.s
(25)  da.d = synthesis(da.d₁, da.d₂, da.predecessor.w, da.successor.w)
(26) else if FLAG₁ == true then
(27)   da.d = da.d₁
(28) else if FLAG₂ == true then
(29)   da.d = da.d₂
(30) end if
(31) end if
(32) da.move(da.d)
(33) if da.predecessor.s >= s_normal then
(34)  da.predecessor.stop(da.predecessor.d)
(35) end if
(36) if da.successor.s >= s_normal then
(37)  da.successor.stop(da.successor.d)
(38) end if
(39) if (da.predecessor.s >= s_normal) && (da.successor.s >= s_normal) then
(40)  da.stop(da.d)
(41) end if
(42) until E(da) <= εγ

Algorithm 2: Connectivity maintenance procedure.

successor. When the transfer completes, the leaving node notifies its predecessor to change successor to the successor of the leaving node. Similarly, the successor of the leaving node is notified to change predecessor to the predecessor of the leaving node. Finally, the leaving node is completely isolated from the Chord network, the successor of its former predecessor is now its former successor, and the predecessor of its former successor is now its former predecessor. Then, the former predecessor of da sets the availability of da to LOGOUT in a monitoring token; the information about the monitoring token is provided later in Section 3.4.

As described above, when the status of a DA changes from normal to low, there is only one function that is removed, namely, the local location optimization. All the remaining functions of a DA are operational in three statuses: normal, low, and alert. The function of global location optimization is removed when the status of a DA changes to serious. Hence, this function tuning algorithm significantly assists the connectivity maintenance. The detailed function tuning algorithm is illustrated in Algorithm 3.

3.3. Directory-Less Mode. In directory-less mode, all DAs in the MANET are stationary. However, they still keep forming a Chord network. Other nodes in the MANET are mobile, namely, UAs and SAs. Though the WSN based on the Chord protocol still exists, there are no topology control and key
searching based on the Chord network. Thus, the directory-less mode is much simpler than the directory-based mode. For DAs and SAs, the service registration is operational as before. This is done to get prepared for a mode switch. The mode switch between the directory-based mode and directory-less mode is detailed later in Section 3.4. In classic directory-less architectures, there are only UAs and SAs. Hence, there is no so called relaying function associated with DAs. The service discovery is only conducted based on service queries issued by UAs and service advertisements issued by SAs. In our model, the DAs provide relaying function for service queries, service advertisements, and service replies. Moreover, since DAs possess a large amount of service information, a DA could issue a service reply to answer a service query based on the service information it has (service information it originally knows and service information it learns through the relaying). For the purpose of mitigating the flooding problem and improving the performance of service discovery, we develop a two-hop zone scheme for the directory-less mode.

3.3.1. Two-Hop Zone Scheme. For a node $n_i$ in the MANET network, the two-hop zone of $n_i$ consists of two parts: one-hop neighbors and two-hop neighbors. A one-hop neighbor $n_{o_{ij}}$ of node $n_j$ is a node which is one hop away from node $n_j$. Though there might be several paths between $n_i$ and $n_{o_{ij}}$, the shortest path between them is one hop. The sufficient and necessary condition for a node $n_i$ to be a one-hop neighbor of node $n_j$ is that they are not able to directly communicate with each other and node $n_j$ is a one-hop neighbor of node $n_i$. One-hop neighbors of node $n_i$ is denoted by $d_{(i,1)}$. Then, we denote the set of one-hop neighbors of $n_i$ by

$$N_{(i,1)} = \{n_{o_{ij}} \mid j \neq i \}.$$  

Similarly, a two-hop neighbor $n_{ij}$ of node $n_i$ is a node which is two hops away from node $n_i$. Though there might be several paths between $n_i$ and $n_{ij}$, the shortest path between them is two hops. The sufficient and necessary condition for a node $n_i$ to be a two-hop neighbor of node $n_j$ is that they are not able to directly communicate with each other and node $n_j$ is a one-hop neighbor of node $n_i$. Two-hop neighbors of node $n_i$ is denoted by $d_{(i,2)}$. The set of two-hop neighbors of $n_i$ is

$$N_{(i,2)} = N_{(i,1)} \cup N_{(j,1)} \cup \cdots \cup N_{(k,l,1)} \setminus N_{(l,i)}.$$  

By the definition of a two-hop neighbor, the number of two-hop neighbors of node $n_i$ can be calculated as

$$d_{(i,2)} = \left| \bigcup_{n_{ij} \in N_{(i,1)}} N_{(j,i)} \setminus N_{(l,i)} \right| - 1.$$  

Note that node $n_i$ is a one-hop neighbor of its one-hop neighbor(s); namely,

$$n_i \in \bigcup_{n_{ij} \in N_{(i,1)}} N_{(ij,1)} \setminus N_{(l,i)}.$$  

Thus, node $n_i$ should be excluded from the calculation of $d_{(i,2)}$.

For an individual node $n_i$ in the MANET network, we denote the frequency of service advertisement of $n_i$ by $d_f$. A service advertisement contains a set of service records. We denote the set of service records contained in a service advertisement by $D(n_i)$, where $|D(n_i)| \geq 1$. During a certain measurement period, the average number of service records sent by node $n_i$ is $\bar{d}_f$ per advertisement. Then, we have $\bar{d}_f \geq 1$. In order to facilitate the presentation of our two-hop zone scheme, we make the following premise.

**Premise 1.** Let $[t_a, t_b]$ be a measurement period, where $t_a < t_b$. We assume that the period is sufficiently long to observe the

**Algorithm 3:** Function tuning procedure.

$$\text{FunctionTuning}(da, e_α, e_β, e_γ)$$

1. if $E(da) > e_α$ then $S = S_i$
2. else if $e_β < E(da) \leq e_α$ then $S = S_l$
3. else if $e_γ < E(da) \leq e_β$ then $S = S_a$
4. else $E(da) \leq e_γ$ then $S = S_i$
5. end if
6. switch (S)
7. case $S_a$: break
8. case $S_l$: $da$.function $\leftarrow da$.function \setminus [LLO]
9. break
10. case $S_a$: $da$.send (ALERT, da.predecessor)
11. $da$.send (ALERT, da.successor)
12. break
13. case $S_l$: $da$.function $\leftarrow \emptyset$
14. $da$.transfer ($\langle$ key, URL $\rangle$, da.successor)
15. $da$.send (CHANGE_SUCCESSOR (da.successor), da.predecessor)
16. $da$.send (CHANGE_PREDECESSOR (da.predecessor), da.successor)
17. break
18. end switch

$$d_{(i,1)} = \left| \bigcup_{n_{ij} \in N_{(i,1)}} N_{(j,i)} \setminus N_{(l,i)} \right| - 1.$$  

$$d_{(i,2)} = \left| \bigcup_{n_{ij} \in N_{(i,1)}} N_{(j,i)} \setminus N_{(l,i)} \right| - 1.$$  

$$d_f = \frac{1}{t_b - t_a} \sum_{t_a \leq t \leq t_b} |D(n_i)|.$$
behaviors of node \( n_i \) together with its one-hop neighbors and two-hop neighbors. Moreover, the behavior characteristics of every node considered remain unchanged, such as the frequency of service advertisement, the frequency of service query, and the average number of service records per advertisement.

Node \( n_i \) receives the service advertisements from all its one-hop neighbors. An update of its local database is carried out according to the service information contained in the received service advertisements. Since the frequencies of service advertisement vary from node to node, the service advertisements from the one-hop neighbors of node \( n_i \) are arrived asynchronously. Moreover, the frequency of service advertisement of a particular node may change from time to time. Hence, the service advertisements from the one-hop neighbors of node \( n_i \) are arrived randomly. For simplicity, we assume that the frequencies of service advertisement of all nodes in the network remain constant during the measurement period \([t_a, t_b]\). In essence, this assumption is given by Premise 1.

We denote the total number of service records received in the service advertisements received by node \( n_i \) by \( sd_{(i,r)} \), and it is computed as

\[
sd_{(i,r)} = \sum_{n_j \in N(i)} d_{(ij)} \cdot \left[ df_{(ij)} \cdot (t_b - t_a) \right].
\] (5)

For each one-hop neighbor of \( n_i \), the number of service records advertised by \( n_j \) is calculated by multiplying the average number of service records per advertisement by the times of service advertisement during the measurement period \([t_a, t_b]\). Premise 1 guarantees that each one-hop neighbor of \( n_i \) performs the service advertisement at least once during the measurement period \([t_a, t_b]\). Thus, the result of \( df_{(ij)} \cdot (t_b - t_a) \) is greater than or equal to 1. For a noninteger number of the times of service advertisement, the floor operation rounds the result to the biggest integer which is less than \( df_{(ij)} \cdot (t_b - t_a) \).

Node \( n_i \) forwards the service advertisements received from other nodes. The total number of service records forwarded by node \( n_i \) is denoted by \( sd_{(i,f)} \). Among the service advertisements received by \( n_i \), whose TTL values are greater than 0 should be forwarded by node \( n_i \). As the two-hop zone scheme we proposed stipulates that the initial TTL value of a service advertisement is 2, the TTL value of each service advertisement received by node \( n_i \) is either 1 or 0. Thus, the coverage of the original service advertisement issued by node \( n_i \) is its one-hop and two-hop neighbors. In other words, the propagation range of an original service advertisement is limited within the two-hop zone of the issuing node. This characteristic could mitigate the flooding problem and reduce the message overhead imposed on the whole network.

Suppose the TTL of the received service advertisements is under Poisson distribution

\[
P\{\text{TTL} = k\} = \frac{\lambda_k \cdot e^{-\lambda_d}}{k!}, \quad \lambda_d > 0, \quad k = 0, 1.
\] (8)

In our two-hop zone scheme, node \( n_i \) is supposed to forward the received service advertisements to all its one-hop neighbors. However, it is necessary that the service advertisements sent to \( n_i \) by its own one-hop neighbor \( n_{ij} \) should not be forwarded back to \( n_{ij} \) by \( n_i \). Though the corresponding one-hop neighbor might vary from one service advertisement to another, we denote it by a node \( n_{iy} \) universally. Then, the total number of service advertisements forwarded by node \( n_i \) is computed as

\[
\lambda_d \cdot e^{-\lambda_d} \cdot \sum_{n_{ij} \in N(i)} \left[ df_{(ij)} \cdot (t_b - t_a) \right] \cdot (d_{(i,1)} - 1).
\] (9)

The universal node \( n_{iy} \) is excluded by the minus one operation. Note that (9) is the number of service advertisements rather than the number of service records. As mentioned above, the number of service records contained in a valid service advertisement is greater than or equal to one. The total number of service records forward by node \( n_i \) is computed as

\[
\sum_{n_{ij} \in N(i)} d_{(ij)} \cdot \left[ df_{(ij)} \cdot (t_b - t_a) \right] \cdot (d_{(i,1)} - 1).
\] (10)

Combining (5) and (10) we have the total number of service records forwarded by node \( n_i \) as

\[
sd_{(i,f)} = \lambda_d \cdot e^{-\lambda_d} \cdot \sum_{n_{ij} \in N(i)} \left[ df_{(ij)} \cdot (t_b - t_a) \right] \cdot (d_{(i,1)} - 1) \cdot sd_{(i,r)}.
\] (11)

Then, (6) could be rewritten as

\[
sd_{(i,a)} = d_{i} \cdot \left[ df_i \cdot (t_b - t_a) \right] + \lambda_d \cdot e^{-\lambda_d} \cdot (d_{(i,1)} - 1) \cdot sd_{(i,r)}.
\] (12)

That is to say, each one-hop neighbor of node \( n_i \) receives service records from \( n_i \).
When node \( n_i \) receives a service query, it checks its local database. If the service query could be answered locally, there is no need to send it out. Otherwise, node \( n_i \) sends it to its one-hop neighbors. Every one-hop neighbor of \( n_i \) will receive a duplicate of the service query. While a service query is traveling through the network, the intermediate nodes will be recorded successively in a travel path. This path information stored in the service query plays an important role for the transmission of a service reply.

We denote the total number of service queries sent by node \( n_i \) by \( ss_{(i)} \), and it consists of two parts:

\[
ss_{(i)} = ss_{(i,j)} + ss_{(i,f)} .
\]  
(13)

Suppose node \( n_i \) issues its own service queries to all its one-hop neighbors at the frequency of \( sf_i \). If the average number of service requests contained in the service queries issued by node \( n_i \) is \( \sigma_i \) per service query, the total number of service requests issued by \( n_i \) is denoted by \( ss_{(i)} \), which is computed as

\[
ss_{(i)} = \sigma_i \cdot [sf_i \cdot (t_{b} - t_{a})] .
\]  
(14)

In our two-hop scheme, we stipulate that a service query contains exactly one service request. Hence, \( \sigma_i \) is identically equal to 1. In the reminder of this paper, we use a service request and a service query interchangeably.

For the service queries received from other nodes, node \( n_i \) checks its local database to determine whether a service query could be answered locally. If a service query could be answered locally, node \( n_i \) will issue a service reply designated to the node which originally issued the service query. The service queries which cannot be answered by node \( n_i \) are disposed according to their TTL values. If the TTL value of a service query is 0, it will be dropped by node \( n_i \), without any forwarding. The service queries whose TTL values are greater than 0 will be forwarded by node \( n_i \). Our two-hop scheme stipulates that the initial TTL value of a service query is 4. Thus, the TTL values of the service queries received by node \( n_i \) are within the set \( \{3, 2, 1, 0\} \). Meanwhile, the coverage of an original service query issued by node \( n_i \) is two one-hop zones. In another word, an original service query is restricted within two one-hop zones of the issuing node. The same as the service advertisement, an upper bound of the initial TTL value of a service query is 4. Therefore, the service queries which cannot be answered by node \( n_i \) are disposed according to their TTL values.

Suppose the TTL value of the received service queries is under Poisson distribution

\[
P\{\text{TTL} = k\} = \frac{\lambda^k e^{-\lambda}}{k!}, \quad \lambda > 0, \quad k = 0, 1, 2, 3.
\]  
(15)

We consider all the service queries sent to node \( n_i \) are directly from its one-hop neighbors. We denote the total number of service queries received by node \( n_i \) by \( ss_{(i,r)} \); it could be calculated as

\[
ss_{(i,r)} = \sum_{n_{ij} \in N_{1,i}} \sigma_{ij} \cdot [sf_{ij} \cdot (t_{b} - t_{a})] .
\]  
(16)

Suppose that node \( n_i \) could be able to answer \( a_i \) percent of the service queries sent to it. The same as the service advertisement, a service query sent to \( n_i \) by its one-hop neighbor \( n_{ij} \) should not be forwarded back to \( n_{ij} \) by \( n_i \). Then, the number of service queries that should be forwarded by node \( n_i \) is obtained analogously:

\[
ss_{(i,f)} = \left( \frac{\lambda_1 e^{-\lambda_1}}{1!} + \frac{\lambda_2 e^{-\lambda_2}}{2!} + \frac{\lambda_3 e^{-\lambda_3}}{3!} \right) \cdot (d_{(i,j)} - 1) \cdot (1 - a_i) \cdot ss_{(i,r)} .
\]  
(17)

Then, (13) could be rewritten as

\[
ss_{(i,j)} = [sf_i \cdot (t_{b} - t_{a})] + \left( \frac{\lambda_1 e^{-\lambda_1}}{1!} + \frac{\lambda_2 e^{-\lambda_2}}{2!} + \frac{\lambda_3 e^{-\lambda_3}}{3!} \right) \cdot (d_{(i,j)} - 1) \cdot (1 - a_i) \cdot ss_{(i,r)} .
\]  
(18)

Compared with service advertisement and service query, the number of service replies is small in nature. In general, the occurrence of a service reply is unpredictable and irregular. It is actually meaningless to impose restrictions on the sending frequency of a service reply. Moreover, in order to improve the performance of service discovery, we do not assign a specific upper bound to the initial TTL value of a service reply. A service reply uses the path information contained in the corresponding service query. The service replies designated to node \( n_i \) are corresponding to the service queries issued by \( n_i \).

We denote the total number of service replies sent by node \( n_i \) by \( sr_{(i)} \); it consists of two parts:

\[
ss_{(i)} = sr_{(i)} + ss_{(i,f)} .
\]  
(19)

\( sr_{(i)} \) denotes the number of service replies corresponding to the service queries which are answered by node \( n_i \) itself. As mentioned above, node \( n_i \) could be able to answer \( a_i \) percent of the service queries sent to it. Hence, \( sr_{(i)} \) could be computed as

\[
ss_{(i)} = a_i \cdot ss_{(i,r)} .
\]  
(20)

Besides sending its own service replies, node \( n_i \) also forwards the service replies designated to other nodes. Among the service replies received by node \( n_i \), those designated to node \( n_i \) will not be forwarded by \( n_i \). For a service reply designated to another node, node \( n_i \) will try to forward it to the next node according the path information contained in the service reply. If the particular next node is not available, the service reply will be dropped by node \( n_i \). Now, we denote the total number of service replies received by node \( n_i \) by \( sr_{(i,r)} \). Suppose that \( h_i \) percent of the service replies received by node \( n_i \) are designated to it. Among the service replies which should be forwarded by node \( n_i \), we assume that \( f_i \) percent of them cannot be delivered to the next node, for the next node might be failed or out of the radio range of node \( n_i \). Then, the total number of service replies forwarded by node \( n_i \) could be calculated as

\[
sr_{(i,f)} = (1 - h_i) \cdot (1 - f_i) \cdot sr_{(i,r)} .
\]  
(21)
Combining (19), (20), and (21), the total number of service replies sent by node \( n_i \) could be rewritten as

\[
 sr_{(i,s)} = a_i \cdot s_{(i,r)} + (1 - b_i) \cdot (1 - f_i) \cdot sr_{(i,r)}
\]  

(22)

3.3.2. Level of Connectivity. In order to facilitate a further investigation of the two-hop zone scheme, we propose a novel method to calculate the level of connectivity for the whole network based on the two-hop zone scheme. In the first place, we formulate the level of connectivity of two arbitrary nodes under the two-hop zone scheme. For two arbitrary nodes \( n_i \) and \( n_j \), the relationship between them is denoted by \( f_k(n_i, n_j) \), where \( k \) indicates the number of hops in the shortest path between \( n_i \) and \( n_j \). As depicted in Figure 6, the shortest path between two arbitrary nodes \( n_i \) and \( n_j \) can be divided into five cases. The intermediate nodes are denoted by \( n_{m_k} \).

In Figure 6(a), node \( n_i \) and node \( n_j \) are able to directly communicate with each other. The number of hops between them is 1. In this case, we have

\[
 f_1(n_i, n_j) = (n_i \in N_{(i,1)}), (n_j \in N_{(j,1)}). \tag{23}
\]

In Figure 6(b), the number of hops between node \( n_i \) and node \( n_j \) is 2. They share the same one-hop neighbor \( n_{m_1} \). Node \( n_j \) is within the two-hop zone of node \( n_i \). In this case, we have

\[
 f_2(n_i, n_j) = (n_i \in N_{(i,2)}), (n_j \in N_{(j,2)}), \tag{24}
\]

\[
 (n_{m_1} \in N_{(i,2)} \cap N_{(j,1)}).
\]

In Figure 6(c), the number of hops between node \( n_i \) and node \( n_j \) is 3. Node \( n_i \) has a one-hop neighbor node \( n_{m_1} \) and a two-hop neighbor node \( n_{m_2} \). Since node \( n_{m_2} \) is a one-hop neighbor of node \( n_j \), \( n_i \) is within the two-hop zone of node \( n_{m_2} \). In this case, we have

\[
 f_3(n_i, n_j) = (n_{m_1} \in N_{(i,3)}), (n_{m_2} \in N_{(j,2)}), \tag{25}
\]

\[
 (n_{m_1} \in N_{(j,2)}), (n_{m_2} \in N_{(j,1)}).
\]

In Figure 6(d), the number of hops between node \( n_i \) and node \( n_j \) is 4. Node \( n_i \) has a one-hop neighbor node \( n_{m_1} \) and a two-hop neighbor node \( n_{m_2} \). Since node \( n_{m_2} \) is a two-hop neighbor of node \( n_j \), \( n_i \) is within the two-hop zone of node \( n_{m_2} \). Node \( n_i \) and node \( n_j \) share the same two-hop neighbor node \( n_{m_2} \). In this case, we have

\[
 f_4(n_i, n_j) = (n_{m_1} \in N_{(i,4)}), (n_{m_2} \in N_{(j,2)} \cap N_{(j,2)}), \tag{26}
\]

\[
 (n_{m_2} \in N_{(j,1)}), (n_{m_2} \in N_{(m_1,1)} \cap N_{(m_3,1)}).
\]

In Figure 6(e), the number of hops between node \( n_i \) and node \( n_j \) is greater than 4. Node \( n_i \) has a one-hop neighbor node \( n_{m_1} \) and a two-hop neighbor node \( n_{m_2} \). Node \( n_j \) has a one-hop neighbor node \( n_{m_4} \) and a two-hop neighbor node \( n_{m_3} \). The dotted line ellipse indicates that nodes \( n_{m_3} \) and \( n_{m_4} \) are connected as the same with node \( n_i \) and node \( n_j \) in the previous four cases, namely, \( f_1 \), \( f_2 \), \( f_3 \), and \( f_4 \). In this case, the relationship between \( n_i \) and \( n_j \) is expressed recursively by

\[
 f_{k+4}(n_i, n_j) = (n_{m_1} \in N_{(i,k+4)}), (n_{m_2} \in N_{(j,k+2)}), (n_{m_3} \in N_{(j,k+2)}), \tag{27}
\]

\[
 (n_{m_4} \in N_{(j,k+1)}), f_{k-4}(n_{m_2}, n_{m_3}),
\]

where \( f_{k-4}(n_{m_2}, n_{m_3}) \) indicates the relationship between \( n_{m_2} \) and \( n_{m_3} \).

More complex situations can be decomposed and modeled through the iteration of different cases illustrated in Figure 6.

The level of connectivity between node \( n_i \) and node \( n_j \) is denoted by \( d(n_i, n_j) \). It is defined as

\[
 d(n_i, n_j) = \begin{cases} 
 0, & k = 0 \\
 1/k, & k = 1, 2, 3, 4 \\
 1 + 1/d(n_i', n_j'), & k > 4 
\end{cases}
\]  

(28)

For \( k = 0 \), there is no path between \( n_i \) and \( n_j \); then we define \( d(n_i, n_j) = 0 \). When \( k > 0 \), the level of connectivity between two nodes monotonically decreases with the increase of the number of hops between them. For \( k = 1, 2, 3, \) and 4, the reciprocal of \( k \) is used as the value of \( d(n_i, n_j) \). For \( k > 4 \), the calculation of \( d(n_i, n_j) \) cannot be done within the two-hop zones of \( n_i \) and \( n_j \). The value of \( d(n_i, n_j) \) is determined recursively. Trivially, a two-hop zone of a node covers two hops. Thus, each time \( k \) is beyond a multiple of 4, another calculation of the level of connectivity is required. Then, the number of recursions could be calculated as \( R = [(k - 1)/4] \).

In the case of \( k > 4 \), node \( n_i'' \) is the particular two-hop neighbor of node \( n_i \) which lies in the shortest path between \( n_i \) and \( n_j \), so do node \( n_j'' \) and node \( n_i \). Taking Figure 6(e) for example, the level of connectivity between node \( n_i \) and node \( n_j \) is computed by

\[
 d(n_i, n_j) = \frac{1}{4 + 1/d(n_{m_2}, n_{m_3})}. \tag{29}
\]
Suppose the set of nodes which node \( n_i \) could communicate with is denoted by set \( N_i = \{ n_1, n_2, \ldots, n_c \} \). Then, we define the level of connectivity for node \( n_i \) as

\[
d(n_i) = \sum_{j=1}^{c} d(n_i, n_j).
\]  

(30)

For \( N \) nodes in the whole network, there exist \( c^2_N = N(N - 1)/2 \) different pairs among them. We define the level of connectivity for the network as

\[
d(N) = \sum_{\forall n_i, n_j \in N, n_i \neq n_j} d(n_i, n_j).
\]  

(31)

Theoretically, a sufficiently large value of \( R \) could result in a situation that there always exists a path between two arbitrary nodes, irrespective of the magnitude of \( N \). With the increase of \( R \), the value of \( d(N) \) will keep increasing until inflection point \((\tilde{d}, \tilde{d})\) is reached. Here, \( \tilde{d} \) is the maximum of \( d(N) \). For \( R \geq \tilde{d} \), \( d(N) \equiv \tilde{d} \). In addition, we define \( \tilde{d} \) as the degree of convergence of the network. The smaller the \( \tilde{d} \) is, the better the network converges.

### 3.4. Autonomic Switch between Two Modes

In order to facilitate the information updating and network status gathering in the WSN, we introduce a monitoring token which is kept passing around the Chord network. The autonomic switch between the directory-based mode and the directory-less mode is based on the network status gathering. By a round, we mean that the monitoring token is transmitted by all the DAs once. The current mode of the service discovery architecture is indicated by a flag called MODE, and its value is DA-BASED or DA-LESS. In addition, the monitoring token contains two lists: the SA list \( sal \) and the DA list \( dal \). The structure of the monitoring token is shown in Figure 7.

![Figure 7: Monitoring token.](image)

Though the unified service information management scheme described in Section 3.1.3 stipulates that all nodes in the MANET have knowledge of the properties and values of services, the DAs initially have no idea of any SAs. Thus, we use the monitoring token to perform the information updating associated with SAs. In general, the appearance of a new SA is accompanied by service registration initiated by the SA. For a DA which received a service registration, it updates local database with the service information. When the monitoring token arrives at the DA, the DA checks whether the SA list of the monitoring token contains the SA of the service registration. If the SA already exists, the DA did nothing for the SA. Otherwise, the DA adds the SA to the SA list of the monitoring token. Meanwhile, the DA checks whether there are unknown SAs. The local database is updated with the information of SAs which are unknown to the DA. That is to say, when the monitoring token arrives at a DA, the DA initiates a two-way update about the information of SAs. When an existing SA is about to leave the MANET, it issues a logout message to an arbitrary DA within its radio range. The DA which receives the logout message first removes the service information associated with the SA in the local database. When the monitoring token arrives at the DA, the DA marks a logout flag in the SA list for the SA. During the transmission of the monitoring token, other DAs become aware of the logout of the SA through the two-way update and remove the service information associated with the SA in their local databases. When the monitoring token arrives at the DA next time, the DA issues a logout acknowledgment to the SA. As long as the SA receives the logout acknowledgment, it could leave the network freely and there is no need to logout again. Although there is no topology control and key searching based on the Chord network in the directory-less mode, the transmission of the monitoring token is still operational. Thus, the two-way update, service registration, and SA logout are also performed normally. The detailed information updating procedure is shown in Algorithm 4.

Besides information updating, the monitoring token also plays another vital function: network status gathering. A DA in the WSN senses the local network status in its vicinity. The information of local network statuses is gathered by the monitoring token, and then a high-level overview of the network status is revealed. This overview of the network status facilitates the autonomic switch between the directory-based mode and the directory-less mode. Suppose our service discovery architecture is under the directory-less mode. If the frequency of service query decreases, the frequency of service advertisement should be decreased accordingly. However, if the frequency of service query keeps decreasing significantly, a switch to the directory-based mode is needed. Now, assume our service discovery architecture is under the directory-based mode; if the frequency of service query increases, the frequency of service advertisement should be increased. If the frequency of service query keeps increasing significantly, a switch to the directory-less mode is needed.

In particular, each time the monitoring token arrives at a DA, if the DA realizes that the frequency of service query it perceived locally is beyond a threshold \( qf_{BAED} \), it adds a plus sign to its entry in the monitoring token. However, if the DA realizes that the frequency of service query it perceived locally is lower than the threshold \( qf_{LESS} \), it checks whether there exists a plus sign in its entry in the monitoring token. A plus sign which already existed will be removed by the DA. Suppose our service discovery architecture is under the directory-based mode. When a DA keeps adding plus signs for several rounds, a local overload is occurring. Moreover, if a considerable number of DAs have encountered a local overload, the directory-less mode should be used. When
the number of DAs which encountered a local overload is small, the directory-based mode should be used. We denote the number of plus signs which indicates a local overload by \( l_o \) (local overload). Moreover, we denote the percentage of locally overloaded DAs which indicates a mode switch between the two modes by \( g_o \) (global overhead). Suppose the service discovery architecture is under directory-based mode. When the monitoring token arrives at a DA, the DA first updates the number of plus signs in its entry. Then, it examines the entries of all DAs. If the percentage of locally overloaded DAs is larger than \( g_o \), it changes the value of MODE in the monitoring token from DA-BASED to DA-LESS for the purpose of informing other DAs to switch their modes. Similarly, a reverse switch from DA-LESS to DA-BASED is triggered by a percentage of locally overloaded DAs which is smaller than \( g_o \). The detailed autonomic mode switch procedure is shown in Algorithm 5.

### 4. Numerical Results

#### 4.1. Experimental Environment and Simulation Parameters.

For the purpose of evaluating our model, we develop a MANET platform which is based on the NS-2 [45]. All simulation results are obtained on HP with Inter Core2 Q9550 2.83 GHz, 4 GB RAM with Debian 2.6.32-48squeeze (Linux version is 2.6.32-5-686) and gcc 4.3.5 (Debian 4.3.5-4).

In our platform, the number of DAs in the WSN is 32. The identifier space of the underlying Chord network is \( 2^m = 2^{16} \). The numbers of SAs and UAs are 25 and 200, respectively. The number of different complete descriptions for profiling services should be \( \prod_{i=1}^{\nu_p} |p_i| \leq 2^{16} \). In [36, 46], the authors use a famous dataset [47] to conduct their experiments. The dataset contains 2507 real web services on the Internet. In this paper, we generate 8000 records of different service information based on the dataset. The consistent hash function based on SHA-1 is employed to calculate the IDs of DAs and the keys of services. The 8000 keys and the corresponding service information (URL, SA) are mapped to the 32 DAs according to the Chord protocol. The IDs of the 32 DAs and the numbers of service records for which each node takes responsibility are shown in Table 3.

For a DA \( da \) and its predecessor \( da.predecessor \) on the Chord network, if the IDs of \( da \) and \( da.predecessor \) are the smallest one and the largest one among all the DAs, respectively, the distance between \( da \) and \( da.predecessor \) is calculated as

\[
dis(da, da.predecessor) = 2^m - da.predecessor.id + da.id.
\]

(32)

Otherwise, we compute the distance between them by

\[
dis(da, da.predecessor) = da.id - da.predecessor.id.
\]

(33)

We define the responsibility ratio of \( da \) as the ratio of the number of service records for which \( da \) takes responsibility
International Journal of Distributed Sensor Networks

Algorithm 5: Autonomic mode switch procedure.

Table 3: Information of the 32 DAs.

| Number | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|--------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|
| ID     | 1456 | 2383 | 2461 | 6511 | 7002 | 7560 | 11438 | 12704 | 13087 | 15018 | 15408 | 20197 | 24126 | 24700 | 33059 | 33837 |
| Records| 562 | 124 | 4 | 514 | 55 | 73 | 468 | 41 | 241 | 34 | 611 | 456 | 71 | 991 | 93 |

| Number | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 |
|--------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|
| ID     | 37271 | 38208 | 39176 | 40539 | 42247 | 43982 | 46340 | 47226 | 47396 | 49090 | 49122 | 52350 | 53654 | 55839 | 59680 | 62300 |
| Records| 430 | 104 | 135 | 174 | 212 | 221 | 278 | 102 | 15 | 218 | 3 | 423 | 126 | 248 | 485 | 335 |

and the distance between da and da.predecessor. And it is computed as

\[
da.rr = \frac{|da.records|}{dis(da, da.predecessor)}.
\]  
(34)

As shown in Figure 8, the responsibility ratios of the 32 DAs are marked as blue crosses and the linear regression equation of them is drawn in red. The responsibility ratios of most DAs are within the range of [0.08, 0.14], except for DA$_3$. The responsibility ratio of DA$_3$ is close to 0.05. The IDs of DA$_2$ and DA$_3$ are 2383 and 2461, respectively. Since, DA$_3$ is close to its predecessor DA$_2$ and the number of service records it is responsible for is small, it takes little responsibility for the service records in the network. As the responsibility is measured by a ratio, it is rational that the responsibility ratio of DA$_{27}$ is larger than DA$_3$.

4.2. Performance Criteria. Though there are various criteria which are used to evaluate the effectiveness of service discovery architectures, we adopt three fundamental criteria: service availability, message overhead, and delay. Each of these criteria is presented below.

4.2.1. Service Availability. Gao and Ma [48] considered that the service availability is the probability that a service is available. It is formulated as the ratio of the number of times a service responds to a user request to the number of total requests made by the user. This definition is targeted at the evaluation of user experience toward a known service. However, the service availability in our model aims to evaluate the performance of service discovery. Thus, we define the service availability from another perspective. In particular, we introduce the service availability of a single node and the service availability of the network. Since service queries and service replies are flowing through all nodes in the network (including UAs, SAs, and DAs), we define the service availability of a single node as the ratio of the total number of incoming service replies to the total number of outgoing service queries. Considering a node $n_i$, the service availability is denoted as

\[
s_{a_i} = \frac{sf_{i}(j,r)}{ss_{i}(j,s)}.
\]  
(35)

Then, the service availability of the network is defined as

\[
s_{a_N} = \frac{\sum_{i=1}^{N} s_{a_i}}{N}.
\]  
(36)
4.2.2. Message Overhead. The concept of message overhead mainly concerns two types of messages: service queries and service replies. In order to improve the performance of service discovery, both the two kinds of messages are duplicated to a certain extent. Specifically, a service query is duplicated for the purpose of reaching more nodes. While a service reply is duplicated in another way, since there are several copies of service queries, it is rational that there may appear several service replies which are originated from various nodes. As service replies are generated in response to the received service queries, we hold that normal service replies should not be considered as message overhead. Thus, we hold that the message overhead should be measured by the portion of service queries which are duplicated redundantly. We denote the number of service queries issued by node $n_i$ by $ss_{(i)}$. The number of the service replies received by node $n_i$ is denoted by $sr_{(i,r)}$. And $b_i$ percent of the service replies received by node $n_i$ are designated to it. We denote the number of duplicates of $ss_{(i)}$ by $dup(ss_{(i)})$. Then, the message overhead of node $n_i$ is defined as

$$mo_i = \frac{dup(ss_{(i)})}{b_i \cdot sr_{(i,r)}}.$$  

Then, the message overhead of the network is defined as

$$mo_N = \frac{\sum_{i=1}^{N} mo_i}{N}. \quad (38)$$

4.2.3. Delay. The response time is a prominent factor of a harmonized user experience. In practice, the delay is greatly concerned by many users. In our model, the delay perceived by a UA consists of two parts: the processing time and the transmission time. Since we do not focus on the relationship between the processing time and the transmission time, we measure the delay in all. Though the number of service queries issued by node $n_i$ is $ss_{(i)}$, it is likely that a part of the service queries finally get no service replies. We denote the number of service queries which finally get one service reply (or several service replies) by $r_j \cdot ss_{(j)}$, where $0 \leq r_j \leq 1$. For a service query which at least has one service reply, the delay associated with the service query indicates the time period between node $n_i$ sends the service query and node $n_i$ receives the first corresponding service reply. We denote the time period by $t$. Then, the delay which node $n_i$ perceived is defined as

$$delay_i = \frac{\sum_{j=1}^{N} r_j \cdot ss_{(j)} t_j}{r_j \cdot ss_{(j)}}. \quad (39)$$

Then, the delay of the network is defined as

$$delay_N = \frac{\sum_{i=1}^{N} delay_i}{N}. \quad (40)$$

4.3. Experimental Results and Analysis. The two characteristics of the directory-based mode are energy conservation and function tuning. Both characteristics concentrate on prolonging the network lifetime for the WSN. In practice, as certain applications require all the nodes are operational, the network lifetime may mean the time period before the first node failure due to energy exhaustion. However, in most applications, the network lifetime means the time period before a number of the nodes in the network exhaust their energy. Notable topology management schemes have been proposed for energy conservation issues, such as [49–51]. However, a major assumption of these schemes is high density of sensors. In order to investigate the performance of our model, we compare it with the approach proposed by Pan et al. [51]. The simulation result shown in Figure 9 indicates a slightly better performance of our model.

As shown in Figure 9, both percentages of the operational nodes for the two models decrease with the increase of time step. In our experiments, we assume that the percentage of failed nodes which indicates the end of the network lifetime is 50%. Note that this percentage is just used for the comparison between our model and Pan et al’s model. In fact, our model is able to operate normally even when there are few nodes left in the network. The percentage of the operational nodes falls below 50% around $t = 275$ in our model. Pan et al’s model performs similar to our model during the period of $t \in [0, 100]$. When $t > 100$, the percentage of the operational modes of Pan et al’s model decreases quickly compared to that of ours. Moreover, the percentage of the operational nodes falls below 50% around $t = 225$ in Pan et al’s model.

Now, we consider our directory-based mode alone. For three fixed values of the frequency of service query $sf_i = 0.006, 0.008$, and 0.01, the service availability of the network and the delay of the network are plotted in Figures 10 and 11, respectively.

As shown in Figure 10, the values of service availability of the network for $sf_i = 0.006, 0.008$, and 0.01 are fluctuated...
in the range of 60%–90%. In our model, the \( (\text{key}, \text{URL}) \) information of a node which leaves the network is transferred to the successor of the node. Thus, the decrease of operational nodes imposes slight influence on the service availability of the network. Note that service queries are generated based on service information which is wishfully created by UAs and DAs. Though the Chord network itself is able to guarantee the normal search of a key, there are chances that some service queries lead to no service replies. This is why the values of the service availability of the network are less than 100% even at the early stage of simulation. Since a large frequency of service query brings more burden to the DAs than that of a small frequency of service query, the slight differences of service availability of the network among the three curves are caused by DAs which are about to fail. These DAs miss issuing service replies by chance. Thus, the ordering of the service availability of the network is roughly \( s_{f_i} = 0.006 > s_{f_i} = 0.008 > s_{f_i} = 0.01 \).

As shown in Figure 11, the values of delay of the network for \( s_{f_i} = 0.006, 0.008, \) and 0.01 increase with the increase of time step. Since the number of operational nodes is decreasing with the increase of time step, the work load which a DA confronts with is getting heavier with the increase of time step. Thus, a service query may encounter a long processing time. Though there are fewer DAs in the WSN, the shortening of transmission time is no match for the growth of processing time. Hence, the total delay is inevitably increased. In addition, a large frequency of service query also results in a larger backlog than that of a small frequency of service query. Thus, the ordering of the delay of the network is roughly \( s_{f_i} = 0.01 > s_{f_i} = 0.008 > s_{f_i} = 0.006 \).

For the directory-less mode, we adopt a fixed setting of \( b_i = 0.1 \) and \( f_i = 0.15 \). Namely, \( b \) percent of the service replies received by node \( n_i \) are designated to it. Among the service replies which should be forwarded by node \( n_i \), \( f \) percent of them cannot be forwarded to the next node. The number of plus signs which indicates a local overload is \( l_o = 4 \). The percentage of locally overloaded DAs which indicates a mode switch between the two modes is \( g_o = 75\% \). Since a sufficiently large value of \( R \) could result in a situation that there always exists a path between two arbitrary nodes, we adopt
themaximumof$k$as10. Then, the number of recursions $R = \lceil (k-1)/4 \rceil = 2$. In order to facilitate an in-depth analysis of our model, we consider two cases of distribution for the level of connectivity for the network: uniform distribution and proportional distribution. For all nodes in the MANET, the numbers of pairs for 10 levels of connectivity are illustrated in Figure 12. In the case of uniform distribution, the numbers of pairs for 10 levels of connectivity are uniformly distributed. In the case of proportional distribution, the number of pairs is in proportion to the level of connectivity.

As previously described, the mode switch between the directory-based mode and the directory-less mode is triggered by the percentage of locally overloaded DAs. Essentially, a local overload for a DA is caused by excessively frequent service queries. In order to investigate the mode switch of our model, we conducted extensive simulations to find an empirical threshold of the frequency of service query. Finally, we learned that the empirical threshold of the frequency of service query is $q_{f_0} = 0.003$. Namely, a value of $q_{f_0}$ which is greater than 0.003 will lead to more than 75% locally overloaded DAs and then triggers a mode switch from directory-based to directory-less.

Figure 13 shows the service availability of the network under the two distributions. When $q_{f_0} \in [0.0005, 0.003]$, the simulation is under directory-based mode. Since the early stage of simulation is under directory-based mode, the difference between the two distributions is not significant. When $q_{f_0} \in [0.003, 0.0055]$, the simulation is under directory-less mode. As the uniform distribution could provide more small levels of connectivity between two nodes than that of proportional distribution, a node could communicate with more distant nodes. Thus, the values of service availability of the network under the uniform distribution are larger than their counterparts under the proportional distribution.

As shown in Figure 14, the delay of the network monotonically increases with the increase of the frequency of service query when $q_{f_0} \in [0.0005, 0.003]$ and monotonically decreases when $q_{f_0} \in [0.003, 0.0055]$. The simulation result indicates that the differences between the two distributions under both directory-less mode and directory-based mode are not significant.

As shown in Figure 15, when $q_{f_0} \in [0.0005, 0.003]$, the values of message overhead of the network under both distributions remain in one. When the mode switches to the directory-less mode, the values of message overhead of the network under both distributions monotonically increase with the increase of the frequency of service query. In addition, the differences between the two distributions under both directory-less mode and directory-based mode are not significant when $q_{f_0} \in [0.003, 0.0055]$. Thus, the
performance of the directory-based mode is superior to the directory-less mode in terms of message overhead.

Through the experimental results shown in Figures 13–15, it can be observed that the mode switch function of our model could significantly improve the performance of service discovery in terms of service availability, message overhead, and delay.

5. Conclusions

With the increasing need for flexible service discovery architectures for service discovery in MANET environment, we propose and evaluate a collaborative self-governing privacy-preserving wireless sensor network architecture. The proposed architecture is based on the Chord protocol and focuses on location optimization and energy conservation. The key function of the architecture is that it provides an autonomic mode switch between the directory-based mode and the directory-less mode. In order to evaluate our model, we developed a MANET platform and conducted extensive simulations regarding three critical criteria concerning service discovery: service availability, message overhead, and delay. The simulation results indicate that the autonomic mode switch between the directory-based mode and the directory-less mode greatly improves the performance of service discovery in terms of the above three criteria. One open issue is how to effectively determine an empirical threshold of the frequency of service query.

Conflict of Interests

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

Acknowledgments

The authors would like to thank the anonymous reviewers for their constructive comments on this paper. This work is supported by Program for the Key Program of NSFC-Guangdong Union Foundation (U1135002), Major National S&T Program (2011ZX03005-002), National Natural Science Foundation of China (60872041, 61072066), the Fundamental Research Funds for the Central Universities (JY10000903001, JY10000901034, K5051203010), and the GAD Pre-Research Foundation (9140A15040210HK61).

References

[1] G. Meditskos and N. Bassiliades, “Structural and role-oriented web service discovery with taxonomies in OWL-S,” IEEE Transactions on Knowledge and Data Engineering, vol. 22, no. 2, pp. 278–290, 2010.
[2] M. Junghans, S. Agarwal, and R. Studer, “Towards practical semantic web service discovery,” The Semantic Web: Research and Applications, vol. 6089, no. 2, pp. 15–29, 2010.
[3] V. A. Santhana and S. R. Balasundaram, “Effective web-service discovery using K-means clustering,” in Distributed Computing and Internet Technology, vol. 7753 of Lecture Notes in Computer Science, pp. 455–464, Springer, Berlin, Germany, 2013.
[4] M. Junghans and S. Agarwal, “Web service discovery based on unified view on functional and non-functional properties,” in Proceedings of the 4th IEEE International Conference on Semantic Computing (ICSC ’10), pp. 224–227, September 2010.
[5] E. Warner, R. Ladner, K. Gupta, F. E. Petry, D. W. Aha, and G. Philip, “System and method for web service discovery and access,” U.S. Patent no. 8,209,407, June 2012.
[6] E. Guttman and J. Veizades, “Service location protocol, version 2,” 1999.
[7] A. Ken, R. Scheifler, J. Waldo et al., Jini Specification, Addison-Wesley Longman, 1999.
[8] Salutation Consortium, Salutation Architecture Specification, The Salutation Consortium Inc., 1999, http://ftp.salutation.org/salute/sa20e1a21.ps.
[9] Microsoft Corporation, "Universal Plug and Play Device Architecture Version 1.0," June 2000, http://upnp.org/index.php/sdcp-and-certification/standards/referenced-specifications/.
[10] Bluetooth Consortium, Specification of the Bluetooth System Core Version 1.0 b: Part E, Service Discovery Protocol (SDP), 1999.
[11] T. D. Hodes, S. E. Czerwinski, B. Y. Zhao, A. D. Joseph, and R. H. Katz, "An architecture for secure wide-area service discovery," Wireless Networks, vol. 8, no. 2-3, pp. 213–230, 2002.
[12] W. Adjei-Winoto, E. Schwartz, H. Balakrishnan, and J. Lillie, "The design and implementation of an intentional naming system," ACM SIGOPS Operating Systems Review, vol. 33, no. 5, pp. 186–201, 1999.
[13] U. C. Koza and L. Tassulas, "Network layer support for service discovery in mobile ad hoc networks," in Proceedings of the 22nd Annual Joint Conference on the IEEE Computer and Communications Societies, vol. 3, pp. 1965–1975, April 2003.
[14] F. Sailhan and V. Issarny, "Scalable service discovery for MANET," in Proceedings of the 3rd IEEE International Conference on Pervasive Computing and Communications (PerCom '05), pp. 235–246, March 2005.
[15] H. Koubaa and E. Fleury, "A fully distributed mediator based service location protocol in ad hoc networks," in Proceedings
[16] M. Klein, B. Königs-Ries, and P. Obreriter, “Service rings—a semantic overlay for service discovery in ad hoc networks,” in Proceedings of the 14th IEEE International Workshop on Database and Expert Systems Applications, pp. 180–185, September 2003.

[17] M. Klein and B. Königs-Ries, “Multi-layer clusters in ad-hoc networkssan approach to service discovery,” in Web Engineering and Peer-to-Peer Computing, pp. 187–201, Springer, 2002.

[18] G. Schiele, C. Becker, and K. Rothermel, “Energy-efficient cluster-based service discovery for ubiquitous computing,” in Proceedings of the 11th Workshop on ACM SIGOPS European Workshop (EW ’11), p. 14, September 2004.

[19] J. Tyan and Q. H. Mahmoud, “A comprehensive service discovery solution for mobile ad hoc networks,” Mobile Networks and Applications, vol. 10, no. 4, pp. 423–434, 2005.

[20] K. Seada and A. Helmy, “Rendezvous regions: a scalable architecture for service location and data-centric storage in largescale wireless networks,” in Proceedings of the 18th International Parallel and Distributed Processing Symposium (IPDPS ’04), p. 218, Santa Fe, NM, USA, April 2004.

[21] S. Sivavakkeesar, O. F. Gonzalez, and G. Pavlou, “Service discovery strategies in ubiquitous communication environments,” IEEE Communications Magazine, vol. 44, no. 9, pp. 106–113, 2006.

[22] C. Lee, A. Helal, N. Desai, V. Verma, and B. Arslan, “Konark: a system and protocols for device independent, peer-to-peer discovery and delivery of mobile services,” IEEE Transactions on Systems, Man, and Cybernetics Part A: Systems and Humans, vol. 33, no. 6, pp. 682–696, 2003.

[23] J. Jeong, J. Park, and H. Kim, “Service discovery based on multicast DNS in IPv6 mobile ad-hoc networks,” in Proceedings of the 57th IEEE Semiannual Vehicular Technology Conference (VTC ’03), pp. 1763–1767, April 2003.

[24] C. Perkins, “IP address autoconfiguration for ad hoc networks,” IETF, 2001. http://www.cs.ucsb.edu/∼ebelding/txt/autoconf.txt.

[25] J. Jeong and J. Park, “Autoconfiguration technologies for IPv6 multicast service in mobile ad-hoc networks,” in Proceedings of the 10th IEEE International Conference on Networks: Towards Network Superiority (ICON ’02), pp. 261–265, August 2002.

[26] M. Barbeau, “Service discovery protocols for ad hoc networking,” in Proceedings of the CASCON 2000 Workshop on Ad Hoc Communications, 2000.

[27] D. Chakraborty, A. Joshi, Y. Yesha, and T. Finin, “Toward distributed service discovery in pervasive computing environments,” IEEE Transactions on Mobile Computing, vol. 5, no. 2, pp. 97–112, 2006.

[28] Z.-G. Gao, X.-Z. Yang, T.-Y. Ma, and S.-B. Cai, “RICFP: an efficient service discovery protocol for MANETS,” in Embedded and Ubiquitous Computing, vol. 3207 of Lecture Notes in Computer Science, pp. 786–795, Springer, Berlin, Germany, 2004.

[29] A. Nedos, K. Singh, and S. Clarke, “Service•: Distributed service advertisement for multi-service, multi-hop manet environments,” in Proceedings of the 7th IFIP International Conference on Mobile and Wireless Communication Networks, 2005.

[30] M. Nidd, “Service discovery in DEAPspace,” IEEE Personal Communications, vol. 8, no. 4, pp. 39–45, 2001.

[31] C. Campo, M. Muñoz, J. C. Perea, A. Marin, and C. Garcia-Rubio, “PDP and GSDL: a new service discovery middleware to support spontaneous interactions in pervasive systems,” in Proceedings of the 3rd IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom ’05), pp. 178–182, March 2005.

[32] C. Lee, S. Helal, and W. Lee, “Gossip-based service discovery in mobile ad hoc networks,” IEICE Transactions on Communications, vol. 89, no. 9, pp. 2621–2624, 2006.

[33] C. N. Ververidis and G. C. Polyzos, “Service discovery for mobile ad hoc networks: a survey of issues and techniques,” IEEE Communications Surveys & Tutorials, vol. 10, no. 3, pp. 30–45, 2008.

[34] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, “Chord: a scalable peer-to-peer lookup service for internet applications,” ACM SIGCOMM Computer Communication Review, vol. 31, no. 4, pp. 149–160, 2001.

[35] Y. Zhang, H. Huang, D. Yang, and H. Zhang, “A hierarchical and chord-based semantic service discovery system in the universal network,” International Journal of Innovative Computing, Information and Control, vol. 5, no. 11, pp. 3745–3753, 2009.

[36] C. Gao, J. Ma, and X. Jiang, “On analysis of a chord-based traffic model for web service discovery in distributed environment,” Journal of Engineering Science and Technology Review, vol. 6, no. 5, pp. 129–136, 2013.

[37] A. Adala and N. Tabbane, “Discovery of semantic Web Services with an enhanced-Chord-based P2P network,” International Journal of Communication Systems, vol. 23, no. 11, pp. 1353–1365, 2010.

[38] V. Mhatre and C. Rosenberg, “Design guidelines for wireless sensor networks: communication, clustering and aggregation,” Ad Hoc Networks, vol. 2, no. 1, pp. 45–63, 2004.

[39] V. P. Mhatre, C. Rosenberg, D. Kofman, R. Mazumdar, and N. Shroff, “A minimum cost heterogeneous sensor network with a lifetime constraint,” IEEE Transactions on Mobile Computing, vol. 4, no. 1, pp. 4–15, 2005.

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

[41] S. Lindsey and C. S. Raghavendra, “PEGASIS: power-efficient gathering in sensor information systems,” in Proceedings of the IEEE Aerospace Conference, vol. 3, pp. 1125–1130, March 2002.

[42] V. Mhatre and C. Rosenberg, “Homogeneous vs heterogeneous clustered sensor networks: a comparative study,” in Proceedings of the IEEE International Conference on Communications, vol. 6, pp. 3646–3651, June 2004.

[43] FIPS Publication, 180-1. Secure Hash Standard, vol. 17, National Institute of Standards and Technology, 1995.

[44] P. Reynolds and A. Vahdat, “Efficient peer-to-peer keyword searching,” in Proceedings of the ACM/IFIP/USENIX International Conference on Middleware, pp. 21–40, 2003.

[45] “The Network Simulator Cns-2,” http://www.isi.edu/nsnam/ns/index.html.

[46] Z. Malik and A. Bouguettaya, “Rateweb: reputation assessment for trust establishment among web services,” The VLDB Journal, vol. 18, no. 4, pp. 885–911, 2009.

[47] E. Al-Masri and Q. H. Mahmoud, “Discovering the best web service,” in Proceedings of the 16th ACM International Conference on World Wide Web (WWW ’07), pp. 1257–1258, May 2007.

[48] C. Gao and J. Ma, “A collaborative QoS-aware service evaluation method for service selection,” Journal of Networks, vol. 8, no. 6, pp. 1370–1379, 2013.
[49] A. Cerpa and D. Estrin, “ASCENT: Adaptive self-configuring sensor networks topologies,” *IEEE Transactions on Mobile Computing*, vol. 3, no. 3, pp. 272–285, 2004.

[50] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris, "Span: an energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks," *Wireless Networks*, vol. 8, no. 5, pp. 481–494, 2002.

[51] J. Pan, Y. T. Hou, L. Cai, Y. Shi, and S. X. Shen, "Topology control for wireless sensor networks," in *Proceedings of the 9th Annual International Conference on Mobile Computing and Networking (MobiCom ’03)*, pp. 286–299, September 2003.
