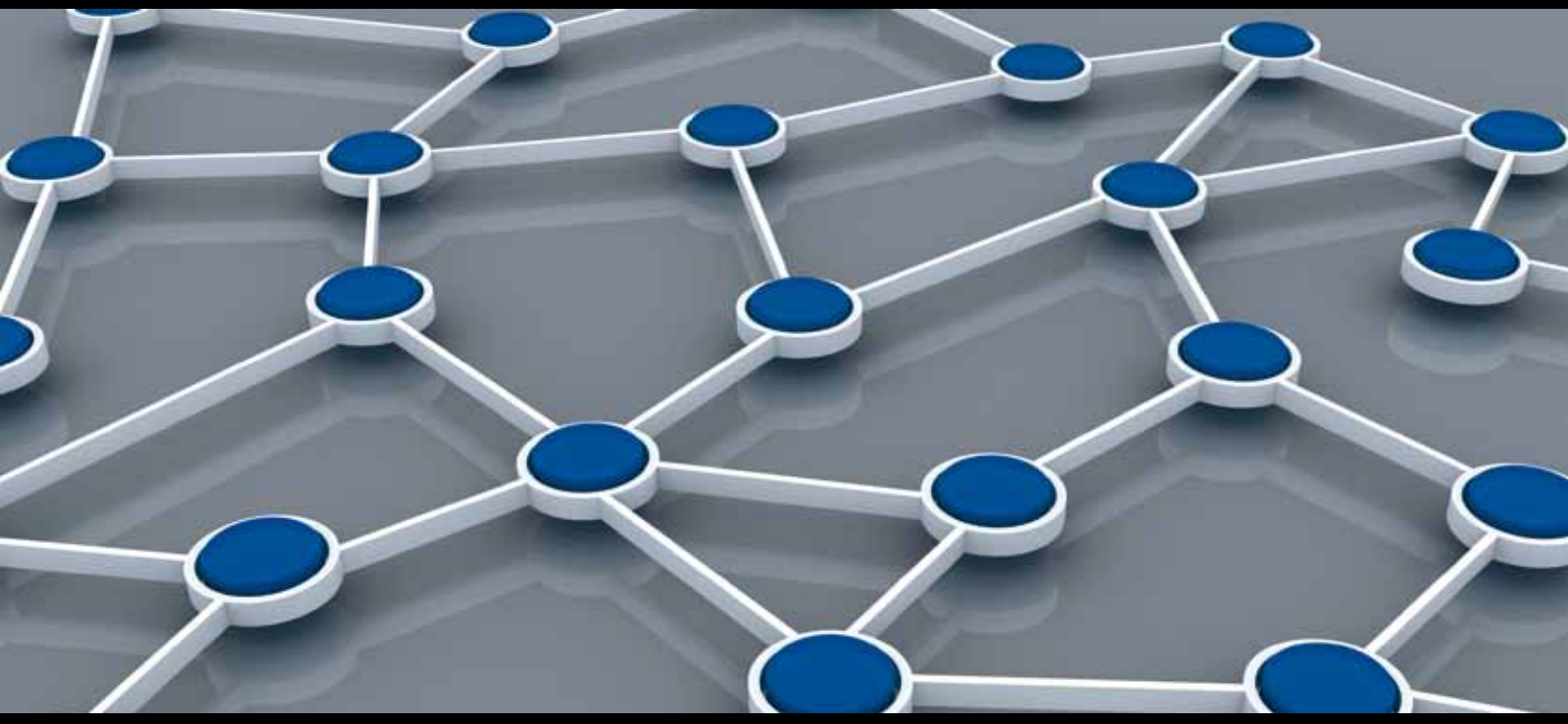
International Journal of Distributed Sensor Networks

Fault-Tolerant and Ubiquitous Computing in Sensor Networks

Guest Editors: Neal N. Xiong, Hongju Cheng, Sajid Hussain, and YanzHen Qu



Fault-Tolerant and Ubiquitous Computing in Sensor Networks

International Journal of Distributed Sensor Networks

Fault-Tolerant and Ubiquitous Computing in Sensor Networks

Guest Editors: Neal N. Xiong, Hongju Cheng, Sajid Hussain, and Yanzhen Qu

Editorial Board

Contents

Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2013, Article ID 524547, 2 pages
http://dx.doi.org/10.1155/2013/524547

Editorial
Fault-Tolerant and Ubiquitous Computing in Sensor Networks

Neal N. Xiong,[1] Hongju Cheng,[2] Sajid Hussain,[3] and Yanzhen Qu[1]

[1] *School of Computer Science, Colorado Technical University, Colorado Springs, CO 80907, USA*
[2] *College of Mathematics and Computer Science, Fuzhou University, Fuzhou, Fujian 350108, China*
[3] *Computer Science, Fisk University, 321 Park Johnson Hall (PJ), Nashville, TN 37208, USA*

Correspondence should be addressed to Naixue Xiong; nxiong@coloradotech.edu

Received 16 September 2013; Accepted 16 September 2013

The wireless sensor networks have already been used in world-wide various applications and are expected to change the future of our world. To connect these heterogeneous networks is still a pioneering issue since people generally require to obtain ubiquitous services anywhere on the earth and ignore the harsh environment where sensors are located. The development of the future sensor networks needs fully new proposals for network architecture varying from the physical layer to upper application layer. The main focus of this special issue will be on the new fault-tolerant and ubiquitous computing issues for large-scale wireless sensor networks. Of course, the selected topics and papers are not an exhaustive representation of the area of fault-tolerant and ubiquitous computing in sensor networks. Nonetheless, they represent the rich and many-faceted knowledge which we have the pleasure of sharing with the readers.

This special issue contains seven papers, where four papers are related to fault-tolerant issue in the distributed wireless sensor networks. Two papers are regarding the media access control and topology controls in network architecture. One paper concerns with the sensor applications and issues in cloud environments.

In the paper, "*Node selection algorithms with data accuracy guarantee in service-oriented wireless sensor networks*," H. Cheng et al. study the node selection problem with data accuracy guarantee in service-oriented wireless sensor networks by exploiting the spatial correlation between the service data. Firstly, they formulate the problem into an integer nonlinear programming problem to illustrate its NP-hard property. Secondarily, they propose two heuristic algorithms for this problem: the previous is designed to select nodes for each service in a separate way, and the second is designed to select nodes accordingly to their contribution increment. And finally, they introduce the simulation results by comparing the algorithms in different environments.

In the paper, "*Energy-efficient bridge detection algorithms for wireless sensor networks*," O. Dagdeviren and V. K. Akram present two distributed energy-efficient bridge detection algorithms for wireless sensor networks. The first algorithm is the improved version of Pritchard's algorithm where two phases are merged into a single phase, and radio broadcast communication is used instead of unicast. The second runs proposed rules on 2-hop neighborhoods of each node and try to detect all bridges in a breadth-first search (BFS) execution session.

In the paper, "*A fault-tolerant method for enhancing reliability of services composition application in WSNs based on BPEL*," Z. Wu et al. present a framework and approach to enhance the reliability of service composition applications in wireless sensor networks. Firstly, they analyze the possible states during the execution of BPEL instance in wireless sensor networks. Secondarily, they present a state framework for modeling execution context in BPEL instance and analyze state transition proposing the state transition models for three types of activities in BPEL instance. Finally, they present a formal approach to model the execution context in BPEL based on this state transition model.

In the paper, "*Distributed fault-tolerant event region detection of wireless sensor networks*," D.-R. Duh et al. present a distributed fault-tolerant event region detection algorithm for wireless sensor networks. The proposed algorithm can identify faulty and fault-free sensors and ignore the abnormal

readings to avoid false alarm. Moreover, every event region can also be detected and identified. The simulation results demonstrate that the algorithm has better performance compared with related works.

In the paper, "*A time slot reservation in modified TDMA-based ad hoc networks with directional antennas*," Y. Li et al. present the bandwidth reservation issue in wireless ad hoc networks with directional antennas and propose a time slot reservation algorithm. The time slot reservation algorithm allows a path reservation with maximal available bandwidth. The performance is analyzed, and simulation results show that it performs better in terms of end-to-end delay, percentage of control packets, and successfully received packets. The scheme is a help to QoS provisioning in wireless ad hoc networks with directional antennas.

In the paper, "*A PSO-optimized minimum spanning tree-based topology control scheme for wireless sensor networks*," W. Guo et al. present a PSO-optimized minimum spanning tree-based topology control scheme NDPSO for the wireless sensor networks by transforming the problem into a model of multicriteria degree constrained minimum spanning tree (mcd-MST). Simulation results show that NDPSO can converge to the nondominated front quite evenly, and the topology derived under the proposed topology control scheme has lower total power consumption, higher robust structure, and lower contention among nodes.

In the paper, "*IM-Dedup: an image management system based on deduplication applied in DWSNs*," J. Zhang et al. study the cloud computing platform for the distributed wireless sensor networks and design an image management system IM-Dedup which uses static chunking (SC) to divide image file into blocks of data and avoid duplication data blocks transmission on network by using fingerprint pre-transmission technology and reduce storage space.

Acknowledgment

We would like to thank the authors for their excellent contributions and patience in assisting us. Finally, the fundamental work of all reviewers on these papers is also very warmly acknowledged.

Neal N. Xiong
Hongju Cheng
Sajid Hussain
Yanzhen Qu

Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2013, Article ID 636797, 12 pages
http://dx.doi.org/10.1155/2013/636797

Research Article

A Time Slot Reservation in Modified TDMA-Based Ad Hoc Networks with Directional Antennas

Yuan Li,[1] Baolin Sun,[1] Xing Luo,[2] and Naixue Xiong[3]

[1] *School of Information Management, Hubei University of Economics, Wuhan Jiangxia District Hidden Dragon Island Development Zone, Wuhan 430205, China*
[2] *722 Research and Development Institute, Wuhan, China*
[3] *School of Computer Science, Colorado Technical University, Colorado, USA*

Correspondence should be addressed to Naixue Xiong; nxiong@coloradotech.edu

Received 30 January 2013; Revised 27 August 2013; Accepted 27 August 2013

Academic Editor: Hongju Cheng

In this paper, we study the bandwidth reservation issue in wireless ad hoc networks with directional antennas. Providing Quality of Service (QoS) support in the wireless ad hoc networks is one of the topics that have received a lot of attention in recent years. QoS provisioning is needed to support multimedia and real-time application in wireless networks. In the Time Division Multiple Accesses (TDMA) frame, the bandwidth is measured by the number of the data time slots. Control slots in the traditional TDMA frame can be used for transmission of the control packets, rather than the transmission of the data packets. We focus on the problem of how to make full use of slots to transmit data packets and do not affect control packets transmission. The purpose is to improve the time slot utilization rate. Based on our proposed modified TDMA frame, a novel time slot reservation algorithm is presented to achieve maximal available path bandwidth. Directional antennas allow transmission energy to be concentrated on a narrow range along a particular direction, which can significantly increase the data rate. The performance of our scheme is analyzed and implemented. Simulation results show that the proposed scheme performs better.

1. Introduction

Future networks will support all types of traffic. Different traffic types need different Quality of Service (QoS) levels for multimedia services and real-time applications Recently, wireless ad hoc networks have had a significant attraction as a complement of terrestrial networks. However, transmitting the realtime and multimedia data packets in such network environment is a very difficult task. The design of a routing protocol for the provision of QoS in a wireless ad hoc network is a challenging task. Wireless channels must be efficiently utilized to improve the QoS support. The limitation in bandwidth and the power of wireless channels challenge the QoS support for wireless networks. The provision of QoS in wireless networks has attracted a lot of researcher's attention [1–7].

In our previous works [4], we have proposed a multipath QoS routing protocol in traditional Time Division Multiple Accesses (TDMA)-based wireless ad hoc networks. This protocol allows transmitting the real-time data packets between a source node and a destination node, and the route between the source node and the destination node can meet the bandwidth requirement of the application. In order to achieve better bandwidth reservation, many studies focus on time slot reservation in traditional TDMA-based wireless networks [4, 5, 8, 9]. These works suppose that nodes are equipped with the omnidirectional antennas.

In this paper, we study the bandwidth reservation issue in wireless ad hoc networks with directional antennas. The bandwidth reservation in wireless ad hoc networks reserves time slots resource in advance in the phase of transmitting the control packets. There are many applications that depend on transmission in wireless networks such as sports telecast. Directional antennas allow transmission energy to be concentrated on a narrow range along a particular direction, which can bring about a higher signal-to-noise ratio (SNR) at the receivers and thus increases the system throughput. In directional antennas, there are two kinds of beamforming

patterns: a lobe pattern and multilobe pattern. In a lobe pattern, only one beam can be used for each transmission. In multilobe pattern, a lot of beams can be used together for one transmission. In the procedure of data transmission, the bandwidth is constrained by the slowest node in the path. Because the transmission in a multilobe splits the transmission power to multiple lobes and then the data rate is decreased by the receiving power, a transmission along a multilobe can bring about a lower data rate than the case of a lobe. There is a challenge in data transmission. If the source node transmits data packets through multiple lobes, it takes less number of transmissions for neighbors to receive the data, but the data rate will be decreased due to the power splitting. On the other hand, if the source node transmits data packets through a lobe, the data rate is high, but it takes more transmissions to cover neighbors. We consider this tradeoff between the number of transmissions and the data transmission rate. The data transmitting problem is an NP-hard problem, and we propose a method to solve the problem.

The problem of our concern is as follows. There is a source node equipped with directional antennas, and a lot of destination nodes that are served by the source node. We suppose that the transmit power of the source node and the beamwidth of directional antennas are fixed. Our object is to find reserved time slots for the source node to destinations such that the paths are optimal. Every path can reserve maximal path bandwidth and minimize the total transmission delay. Our work allows for the adjustment of beam orientations and the use of multilobes. Some research has been done on the topic of data transmitting using directional antennas, such as [10–13]. These research works either use a lobe pattern or use a fixed beam orientation. Some research considers broadcasting scheduling [14], while our research considers the paths from the source node to several destination nodes with reserved time slots. At the same time, our scheme allows using the multilobes and beam orientation adjustment.

In wireless networks, nodes transmit data packages by using an omnidirectional antenna that radiates its power equally in all directions. However, directional antennas allow a node to transmit data in a particular direction. At the same time, a receiving node can focus its antenna on a particular direction. This antenna model provides the following advantages [8]: (1) a smaller amount of power can be used; (2) it can increase the spatial reuse; (3) route has shorter hops and smaller end-to-end delay. In this paper, we present a novel time slot allocation scheme in wireless networks with directional antennas by modifying traditional TDMA frame. We begin from a traditional TDMA frame structure. Then, we propose the modified TDMA frame structure mode. On the basis of it, a bandwidth allocation scheme by using directional antennas is presented.

The time slot reservation problem is an NP-hard problem. We proposed a three phases to solve this problem. Firstly, we use a lobe pattern to all nodes by adjusting the beam orientations. Secondly, we proposed a time slot reservation algorithm for paths from source to destinations. Lastly, based on the results from the first phase and second phase, a multilobes pattern is presented to minimize the total transmission delay. The solution for every phase is optimal. Extensive simulations are conducted to evaluate our proposed method.

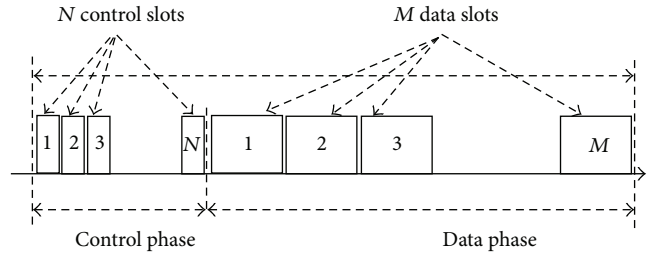


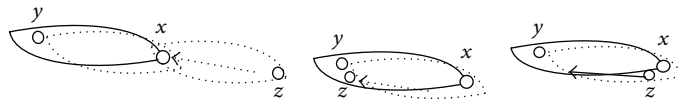
Figure 1: Traditional TDMA frame structure model.

This paper is organized as follows. In Section 2, we discuss the related works with our search. Section 3 presents the modified TDMA frame structure mode and the solved problem description. In Section 4, we provide our time slot reservation scheme. Some simulation results are provided in Section 5. Finally, the paper is concluded in Section 6.

2. Related Works

In the wireless networks, how to make use of resources is dependent on the Medium Access Control (MAC) layer protocol. There are two main MAC methods for the wireless networks which are TDMA MAC layer protocol and IEEE 802.11 MAC layer protocol. The 802.11 MAC layer protocol cannot guarantee the real-time communication because it is a contention-based approach. The TDMA MAC layer protocol is a schedule-based approach by reserving time slots for nodes. In order to achieve the bandwidth resource reservation of a selected route in the TDMA-based ad hoc networks, many time slot reservation algorithms have been proposed [8, 15, 16]. These slot reservation algorithms in TDMA-based networks can avoid slot reservation conflict. In this paper, we adopt the TDMA MAC layer protocol. In the traditional TDMA frame structure model, TDMA superframe consists of the control phase and the data phase [8]. The control slots in control phrase are used by transmitting control packets, such as route request packets, route response packets, and route update packets. Every node in the wireless network has its control slot in the control phase. The data slots in data phase are used by transmitting data packets. Figure 1 shows the traditional TDMA frame structure model.

Many QoS routing protocols adopt the TDMA frame structure as the MAC layer. Hsu et al. propose a bandwidth reservation for QoS routing protocol in [15], which supposes that all nodes are equipped with omnidirectional antennas. In [16], Du considers the efficient QoS bandwidth provisioning in hybrid wireless networks. These bandwidth reservation schemes also suppose that nodes use the omnidirectional antennas. The data packages are broadcasted to all neighbor nodes in all directions. In the resource scheduling scheme in [8], Jawhar and Wu suppose that nodes transmit data by using directional antennas. They have presented three time slot allocation conditions. A slot t is free and can be reserved to send data from node x to neighbor y if the following three



(a) Example of condition 1 (b) Example of condition 2 (c) Example of condition 3

Figure 2: Transmission with directional antennas.

conditions are satisfied [8]: (1) Node x do not receive data in slot t at any antennas, and neighbor node y do not send data in slot t at any antennas; (2) Neighbors of x do not receive data in slot t, from x where neighbors are in the same direction as receiver y; (3) Neighbors of receiver y do not send data in slot t, from y where neighbors are in the same direction as transmitter x. These time slot allocation conditions can avoid the slot reservation conflict problem.

Figure 2 shows the examples of transmission with directional antennas. According to condition 1, Figure 2(a) shows that node x will not transmit to neighbor y at the slot t, because node x is receiving data at the slot t. According to condition 2, the neighbor node z of the node x is receiving at the slot t in Figure 2(b). Node x will not transmit to neighbor y because y and z are in the same direction of x. In Figure 2(c), node z is sending at the slot t according to condition 3. Node y will not receive from x because x and y neighbor z are in the same direction of y.

There are other slot allocation schemes by using directional antennas technology. Bazan and Jaseemuddin [17] propose the routing and admission controls for wireless mesh networks with directional antennas. Authors study the problem of bandwidth guaranteed routing in contention-based wireless mesh networks with directional antennas. Chen et al. [18] present a shoelace-based QoS routing protocol for mobile ad hoc networks using directional antennas. This scheme offers a bandwidth-based routing protocol for QoS support by using the concept of multipath. Chin et al. [19] propose a novel spatial TDMA scheduler algorithm for transmitting/receiving in wireless mesh networks. Li and Luo [20] present a slot allocation scheme based bandwidth and delay in satellite networks using directional antennas. Feng et al. [21] present a QoS constrained cognitive routing scheme based on directional antennas. Liu et al. [22] consider the topology control for multichannel multiradio wireless mesh networks using directional antennas. Chang et al. [14] propose a minimum delay broadcast scheduling for wireless networks with directional antennas.

The above research works take into account directional antennas. Directional antennas allow transmission energy to be concentrated on a narrow range along a particular direction, which can significantly increase the data rate.

The issue of transmission using directional antennas has been firstly discussed on the physical layer. Sidiropoulos et al. [23] proposed the idea to maximize the smallest signal-to-noise ratio over all the nodes subject to a bound on the transmit power by employing semidefinite relaxation techniques. Similar research joint with power control can be found in [24, 25]. These research works discussed the problem from physical layer without considering the beam scheduling for several transmissions. There are a few studies about transmission with power control using directional antennas at higher layer [10–12]. Aiming to minimize the total transmission delay, Sen et al. [10] presented a two-step solution. In [11], Sundaresan et al. provided two models for power allocating. Equal power split model (EQP) splits the power equally among all the lobes, while asymmetric power slit admits power adjustment among the lobes. Based on the two models, two greedy algorithms were presented to get minimum transmission delay. Zhang et al. [12] resolve the challenge of beam combination discussed in [11] with the similar objective under two models. They achieved an optimal solution under EQP model and showed better performance compared with the methods in [11]. None of the above research works tak the orientation adjustment and multilobe transmission together into account in multipath beam scheduling. Because they all assume that the nodes use fixed beam orientations, the performance improvement of the above algorithms is restricted.

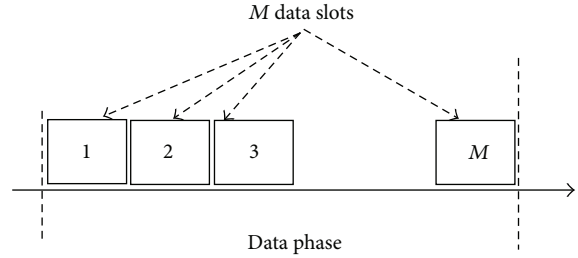


Figure 3: Modified TDMA frame structure model.

3. Model Improvement

3.1. Modified TDMA Frame Structure Mode. In the traditional TDMA frame model showed in Figure 1, control packets are transmitted in control phrase. Control slots in control phrase are not to be used by transmitting of data packets. In order to save the slots consumption of control packets, we will modify the traditional TDMA frame structure model by deleting the control phrase from the TDMA frame.

Figure 3 shows the modified TDMA frame structure model. Compared with traditional TDMA frame, it reduces the control phrase. Suppose that the control packet is transmitted in the data slot of data phase. This data slot will be reserved if node receives the response control packet. After

that, the data packets will be transmitted in the reserved data slot.

In the modified TDMA frame model, a TDMA frame only consists of a lot of data slots. Every data slot is used to transmit data packets, control packets, or both, which depends on the node's need. The data packet can be transmitted with the control packet when required. By adopting special slot reservation algorithm with directional antennas in the modified TDMA frame, it can avoid data packets conflict in transmission.

3.2. System Description. The system consists of a source node equipped with directional antennas and a set of destination nodes with directional antennas. The source node wants to send data packets to destination nodes through intermediate nodes. The destination nodes are denoted by $V = \{v_1, v_2, \ldots, v_n\}$. The beamwidth q is denoted by the angle of directional antennas. The beamwidth q is fixed, and $q < \pi/2$. The source node has fixed transmission power P_t. The location of neighbor node n_i of the source node is denoted by the (d_i, θ_i) centered from the source node. d_i is the distance from the source node to the neighbor node n_i, and θ_i is the angle of the link from the source node to the node n_i from positive horizontal direction. When the source node wants to send data to destination nodes, it tries to find a proper reserved bandwidth path to every destination node such that paths total delay is minimized. When the source node transmits data to neighbor n_i by using a lobe whose beamwidth is q, the power received by n_i can be computed by using Friis Transmission Formula [26]:

$$P_i = \frac{2\pi P_t}{d_i^w q}, \tag{1}$$

where w is the coefficient of path loss, which is usually between 2 and 4. Because the beamwidth q is usually narrow and the energy is concentrated, once a node is covered by a lobe, we suppose that receiving power of the neighbor node is determined by its distance to the source. In (1), we suppose that the gain of signal by using directional antennas is $2\pi/q$.

If we use multilobe pattern, we suppose that the power is equally split among the multiple lobes. We assume that there are n transmissions required for the source node to send a data packet to n destination nodes. Let $S_1, S_2, \ldots, S_k$ denote k sets of neighbors, and let every set denote one transmission. We suppose that N_j denote the number of lobes on jth transmission, and the receiving power of neighbour node $n_i \in S_j$ satisfies the following formula:

$$p_{(i,j)} = \frac{P_i}{N_j}, \tag{2}$$

which means that the transmit power for each lobe in multilobes mode is $1/N_j$. We assume that the maximum available bandwidth $B_{(S,i)}$ of neighbor node n_i to receive packet from source node in jth transmission. $B_{(S,i)}$ is determined by the received signal-to-noise ratio (SNR) of node n_i, which can be computed by the following formula:

$$B_{(S,i)} = f\left(\mathrm{SNR}_{(S,i)}\right), \tag{3}$$

where $f()$ is a mapping function from SNR to bandwidth of node n_i and the following formula is computed:

$$\mathrm{SNR}_{(S,i)} = \frac{P_{(S,i)}}{E}, \tag{4}$$

where E is the environmental signal noise. According to formula (3) and formula (4), we can get the following formula:

$$B_{(S,i)} = f\left(\frac{P_{(S,i)}}{E}\right), \tag{5}$$

which means that the bandwidth of link from the source node S to n_i. n_i is a neighbor node of S. In data transmission duration, the bandwidth of a path from the source node S to the destination node $v_i \in V$ is determined by the lowest bandwidth of links in path. Let B_{path} denote the path bandwidth of a path from S to node v_i. Suppose that the $B_{(i,j)}$ is the bandwidth of the link (n_i, n_j) in the path. The Following formula can be calculated:

$$B_{\mathrm{path}} = \min_{n_i n_j \in \mathrm{path}} \left\{B_{(i,j)}\right\}, \tag{6}$$

which means that the path bandwidth is the lowest link bandwidth of the path. $n_i n_j$ denotes a link from node n_i to node n_j in the path from the source node S to the destination node v_i. The delay $l_{(i,j)}$ for the link (n_i, n_j) is calculated by the formula where L is the size of a data packet:

$$l_{(i,j)} = \frac{L}{B_{(i,j)}}. \tag{7}$$

The delay l_{path} of a path is calculated that formula (8), which means that the delay of a path is the sum of all the link delay of the path. If the source node can receive route response packets from n destination nodes, the total delay of multiple paths can be described as in formula (9):

$$l_{\mathrm{path}} = \sum_{n_i n_j \in \mathrm{path}} l_{(i,j)} = \sum_{n_i n_j \in \mathrm{path}} \frac{L}{B_{(i,j)}}, \tag{8}$$

$$D = \sum_{1 \le i \le n} l_{\mathrm{path}_i}. \tag{9}$$

We will study how to adjust orientations of directional antennas for source node to form a set of multilobes such that the total transmission delay defined in (9) is minimized. The minimum delay broadcast scheduling problem is proved to be an NP-hard problem [14], and we will prove that the minimum delay scheduling of slot reservation in multiple paths from the source node to multiple destination nodes is also an NP-hard problem. We prove the theorem according to the weighted set cover problem, a well-known NP-hard problem [27]. The weighted set cover problem is described as follows. Let $A = \{1, 2, \ldots, n\}$ express a set of n elements. B consists of finite sets $B_1, B_2, \ldots, B_n \subseteq A$ with weight w_j for $B_j \in B$. We denote $\cup(B_j : 1 \le j \le n)$ by A. The objective is to find a subset of B that covers all elements in A with a minimum weight. According to the weighted

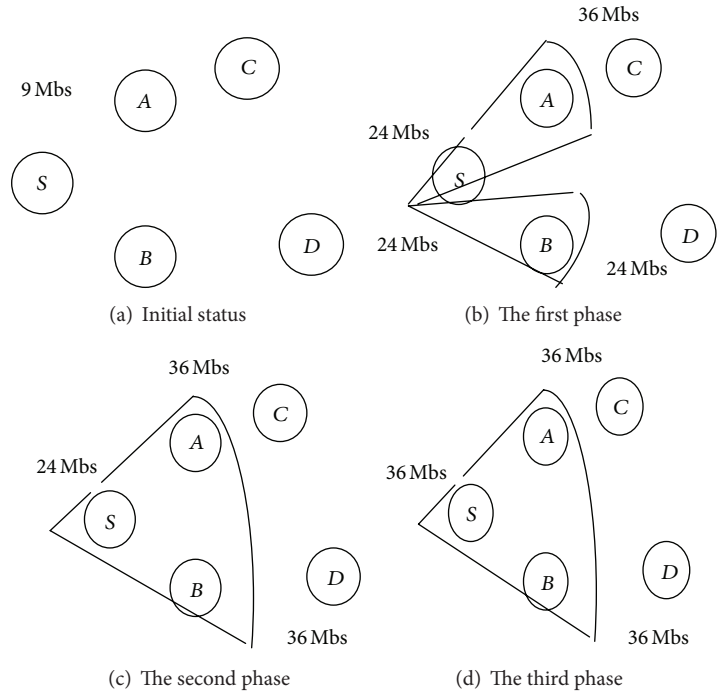


Figure 4: An example of three phase method.

set cover problem, let every element $i \in A$ correspond to the destination node v_i. B_j denotes the set of nodes covered by one transmission and w_j denotes the corresponding transmission delay. Our problem is to find an optimal set of transmissions to cover all destinations to minimize the total transmission delay, which is just to find a subset of B with minimum weight in weighted set cover problem. General greedy approach for weighted set cover problem cannot be used for our method. We will use a three-phase method to solve the problem.

3.3. A Three Phase Scheduling Method. The problem we study is a complicated optimization problem. The complexity of this problem comes from the following three concurrent cases. (1) Which subset of neighbor of the source node may be covered by a lobe? By adjusting the beam orientation, the bandwidth of this lobe varies because the bandwidth is constrained to the smallest bandwidth of the neighbors in this lobe. (2) Which subset of lobes may form a multilobe? Once more lobes form a multilobe, the number of transmissions reduces. However, the bandwidth of transmission reduces due to the power scatter. Once less lobes form a multilobe, the bandwidth for each transmission increases and the number of transmissions increases. (3) Which lobes may be used by the nodes in the path such that the available path bandwidth is maximized?

Based on the above three subproblems, we present a three-phase scheduling method. In the first phase, we only adopt a lot of lobes to cover all neighbors of the source node such that the transmission delay is minimized. In the second phase, we group multiple lobes to form multilobes to minimize the delay. Figure 4 shows an example. We suppose that the size of the data packet is 1 Mb and the bandwidth of the path is 9 Mbps. Suppose that there are two paths from the source node to two destinations which the hop number is two. If all nodes use omnidirectional antennas to transmit the data packets, the transmission delay is 0.22 s which is the result of 1 divided by 9 times 2. By the end of the first phase, two lobes are adopted, which means that the source node can transmit two times to cover two neighbors to transmit data packets to two destinations. The total delay is 0.15 s. After the second phase, the source needs one transmission for covering two neighbors and the total delay for two destination nodes is 0.11 s. In the third phase, we adopt time slot scheme described in Section 4. The total delay for two destination nodes is 0.108 s, and the total bandwidth is 36 Mbs.

4. Our Time Slot Reservation Scheme

In wireless networks, node mobility is handled by taking into account the slot allocation modes of nodes in two hops area when requesting to reserve slots for multimedia application. Therefore, it can avoid collisions with the neighbors in two hops when a node moves. When a node starts to move, the nodes that will become its neighbors are those which were its two-hop neighbors. In that case, there is no collision because these neighbors have reserved different data slots. There are three types of control packets which are route request packet, route response packet, and update packet. The route request package is used to search route and request slot allocation. The route response package is used to confirm the route and reserve the allocated slots. The update packet is used to inform the neighbors in the two-hop area about the reservation status change.

```
if (intermediate node i receives a route request package at the first time)
{  do {
       if (direction from node i to the next hop is similar to the direction of reserved slot in two-hop area)
          it chooses a free slot for node i.
       else it chooses the slot which is reserved in two-hop area with different direction.
   } while (no collision is detected in the selected slot)
   if (number of selected slots at node i satisfies the bandwidth requirement for the application)
      it forwards a request package in the allocated slot after node i and slots are added to
      the reservation table.
   else node i discards the route request package.
} end if
```

Algorithm 1

```
if (node i receives a route response package at the first time)
{  do {
       it modifies the status of slots from allocated to reserved in the reservation table, and it adds
       the antenna direction of reserved slot to the reservation table;
       it broadcasts the update packet to neighbors of node i in reserved slot;
   } while (there is an allocated slot in reservation table)
} end if
```

Algorithm 2

In the wireless ad hoc networks with directional antennas, we suppose that each node is equipped with directional antennas. It is assumed that a multibeam antenna arrays (MBAA) is capable of broadcasting by adjusting the beam width [8]. Therefore, in order to consider antennas' beam direction for the slot allocation-based directional antennas, node mobility issue tackling is based on reserving slots in the two-hop neighboring and corresponding angular directions of antennas.

We suppose that each node maintains a slot reservation table which contains four columns. The first column means the node ID that has reserved a time slot; the second column presents the number of reserved slots in the modified TDMA frame; the third column records reserved node which is a direct neighbor or a two-hop neighbor or the node itself (expressed by 1, 2, or 0). The fourth column indicates the antenna direction of reserved slot. A node can select a slot which is reserved by its two-hop neighbors when the antenna directions of slot are not the same.

In our proposed time slot allocation scheme-based directional antennas, we consider the two-hop neighbor's slot reservation and antenna directions. The time slot allocation scheme can be described with Algorithm 1.

After the node selects slots and meets the bandwidth requirement, the route request package will be forwarded to neighbors in the allocated slots. The allocated slots are recorded in the slot reservation table of node i, but the status of the slot is allocated not reserved. When the node i receives a route response package, following slot reservation, Algorithm 2 is described.

When the intermediate node i reserves the slots and broadcasts the update packet, neighbors j precedes the operation shown in Algorithm 3.

When the source node receives the several route response packages from the same destination, it chooses the smallest delay path according to formula (8) in Section 3.2. After determining all paths to all destination nodes, the source node starts to transmit data packets in reserved slots along the selected paths because the reserved smallest delay paths has been found. The nodes in other reserved paths release reserved slots after receiving route release packets from the source node. The intermediate nodes forward the data packets in reserved slots once they receive the data packets. When the intermediate node does not receive a data packet in the reserved slot, it means that the packet conflicted due to mobility of nodes. In order to avoid the packet conflict due to the mobility of the nodes, Algorithm 4 is performed.

If the source node wants to transmit the data packets to multiple destination nodes, it chooses the smallest delay multipath according to formula (9) in Section 3.2. In order to describe the details of the time slot allocation and reservation scheme, we will start by explaining the one-hop functioning. Then, node behavior in package conflict case will be described. Finally, we will analyze the two-hops functioning. Supposing that the node in a dense ad hoc network will find neighbors in each $2R$ area, where R is a minimum transmitting range of the nodes.

4.1. One-Hop Functioning Case of the Slot Allocation and Reservation. The principle of our slot allocation and

if (node *j* receives an update packet from its neighbor *i*)
{ it adds a row to its reservation table, which indicates that neighbor node *i* has reserved slots at antenna direction;
it informs its neighbors about the updates in its reservation table;
} **end if**

Algorithm 3

if (intermediate node *j* does not receive a data packet from its neighbor *i* in reserved slot)
{ it deletes a row, which indicates that the node *i* and corresponding reserved slots will be deleted from its reservation table;
it deletes a row, which indicates that the node *j* itself and corresponding reserved slots will be deleted from its reservation table;
it informs its neighbors about the updates in its reservation table by broadcasting the update packets;
neighbors perform the Algorithm 3, which is modified to delete a row in reservation table;
} **end if**

Algorithm 4

reservation approach is based on the modified TDMA frame structure model for the subsequent data transmission. In the traditional slot reservation algorithm [18], there is no obvious difference between slot allocation and slot reservation. In our scheme, when the node receives the route response package, allocated slots will be changed to reserved status. If the node does not receive the route response package in the following frames, allocated slots will be changed to free status in order to free more slots for other route request packages.

In the reservation table, we add a row without antenna direction information when the node receives a route request packet and meets the conditions of Algorithm 1. The selected slots are added to the reservation table, but antenna direction information is not added to the table. The reason is that these selected slots are allocated status. Only when the node receives a route response package, the status of selected slots are changed from allocated to reserve. Therefore, the antenna directions of reserved slots are added to the reservation table.

If the node that has allocated slots does not receive a route response package in subsequent three frames, the status of selected slots is changed from allocated to free. In other words, the row without antenna direction information will be deleted from the reservation table. If the node that has been reserved slots does not receive a data packet in reserved slots of subsequent frames, the status of reserved slots will be changed to free. Therefore, the row with antenna direction information will be deleted too. The slots with free status will be ready for the subsequent route request packets.

In order to illustrate the one-hop functioning case of the allocation and reservation scheme-based directional antennas with four beams, we suppose that there are four nodes in Figure 5. At first, the slot reservation tables of four nodes are empty, which means that all slots are free in four nodes' neighborhood. The line between two nodes indicates the neighboring. Table 1 shows the reservation table of node, which contains node ID, reserved slots set, direct

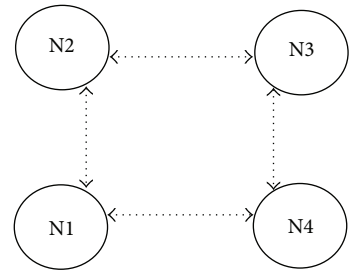


Figure 5: Neighboring nodes in one-hop functioning case.

Table 1: Reservation table of node N1.

Node ID	Slots	Direct node	Direction

neighbor node, and antenna direction. When node N1 wants to transmit data packets to neighbor node N4, the following steps will occurr. In this example, we focus on the allocation and reservation of the free slot 1 for transmission from N1 to N4, which are one-hop neighbor.

Step 1 (during the slot 1 of the first modified TDMA frame). N1 starts to broadcast a route request packet in free slot 1. After neighbors N2 and N4 receive this request packet, these nodes add a row to their reservation tables. The triple (N1, S1, 1) is added to the reservation tables of N2 and N4. The triple (N1, S1, 0) is added to the reservation table of N1. It is notable that the antenna direction is not added to the reservation tables. The reason is that the status of slot 1 is allocated.

Step 2 (during the slot 2 of the first modified TDMA frame). N4 starts to broadcast a route response packet in free slot 2. After neighbors N1 and N3 receive this route response packet, node N1 adds antenna direction of reserved slot to the

reservation table. The reservation table contains a row (N1, S1, 0, 0), which means that N1 reserves slot 1 at the horizontal 0 degree direction.

Step 3 (during the slot 1 of the second modified TDMA frame). In the subsequent modified TDMA frame, N1 transmits data packet during the slot 1. The update packet can be piggybacked with the data packet in the slot 1 of the second TDMA frame. After neighbors N2 and N4 receive this update packet, they add the antenna direction to their reservation tables. In other words, a row (N1, S1, 1, 0) records the slot reservation information of neighbor N1.

If N1 want to transmit data packets to neighbor N2 after N1 transmitted data packets to neighbor N4, slot 1 in the reservation table will be selected again. The reason is that 90 degree direction of the link (N1, N2) is different from 0 degree direction of reserved slot 1 for link (N1, N4) when using directional antennas with four beams. Therefore, a row (N1, S1, 0, 90) is added to the reservation table of N1, which means the slot 1 is reserved two times in two different antenna directions. N1 will transmit data packets to two neighbours in the same slot without conflict with each other because of the use directional antennas. If using omnidirectional antennas, N1 cannot transmit data packets to two neighbours in the same slot. N1 must select a free slot to reserve for the new transmission in order to avoid slot reservation conflict. From the third modified TDMA frame, node N1 can transmit data packets to N2 and N4.

Step 4 (during the slot 1 of the fourth modified TDMA frame). Suppose that N2 and N4 find that they do not receive data packets during the slot 1. There is a transmission break problem due to mobility of node, which brings about N2 and N4 are not one-hop neighbour of the node N1. In this case, N2 and N4 delete a row in their reservation tables, which means that the status of slot 1 is changed from reserved to free. The free slot 1 may be allocated for other route request packets in the subsequent frame.

4.2. The Collision Case. A collision case will be found by the neighbor nodes which broadcast the route request packets. The neighbors will notice that reservation is canceled when no data packet arrives in the reserved slot of the subsequent modified TDMA frame. In order to decrease the packet collision instances, the node which has caused a collion will not reselect the same slot to broadcast a route request packet. In order to illustrate the collision case, we consider the following case shown in Figure 5. We suppose that the modified TDMA frame contains 10 data slots, and slot 1 was, respectively, reserved by node N1 at horizontal and vertical directions for transmitting data packets to neighbour N4 and N2. When two neighboring nodes N2 and N3 broadcast the route request packets during the same slot 2 after N1 starts to transmit data packets, the following steps will be performed.

Step 1 (during the slot 2 of the first frame after N1 starts to transmit data packets). There is a collision when N2 and N3 broadcast route request packets during the same slot 2. Because N2 and N3 send request packets in the same time, a collision is found and the slot allocation algorithm is not performed by the nodes N2 and N3 as well as their neighbor N1 and N4. Node N2 calculates the slot ID from which it will search a free slot to use for sending route request packet. The slot ID is calculated according to the total number of slots in modified TDMA frame modulo the ID of node. In Figure 5, free slot 2 is allocated to node N2, because 10 modulo 2 equals 0. The first free slot that starts from 0 is slot 2, because slot 1 has been reserved for link (N1, N2).

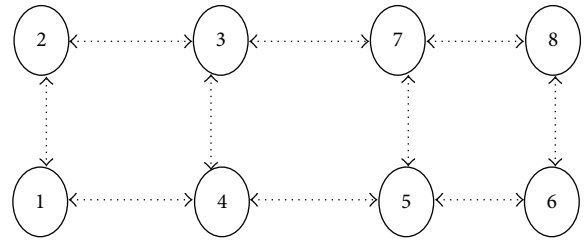


Figure 6: Neighboring nodes in multiple-hops functioning case.

Step 2 (during the slot 3 of the first frame after N1 starts to transmit data packets). Node N3 calculates the slot ID from which it will search a free slot to use for sending route request packet. The free slot 3 is allocated to N3, because 10 modulo 3 is equal to 1. The first free slot that starts from slot 1 is slot 3, because slot 1 and slot 2 have been reserved. Nodes N2 and N3, respectively, broadcast the route request packets in different slots. Therefore, there is no collision case between N2 and N3.

4.3. Multihops Functioning Case of the Slot Allocation and Reservation. It is not sufficient to consider the slot allocation and reservation in one-hop neighboring when the route request packet traverses multiple hops in mobile ad hoc networks. If a two-hop neighbor node starts to move suddenly, it may come to a 1-hop neighbor node or three-hop neighbor. The reservation information of the two-hop neighbor node must be taken into account. The two-hop neighbors may reserve the same slot with the source node before becoming a one-hop neighbor. Therefore, the collision can take place because neighboring nodes reserve the same slot. The slot reservation table must consider the reservation of all two-hop neighboring nodes. In order to illustrate the reservation procedure in the multiple hops functioning, we suppose that there are eight nodes in Figure 6. Initially, node N1 has reserved the slot 1 for the link (N1, N4) at the horizontal direction. In this example, we will describe the following steps that will occur when node N3 needs to transmit data packets to node N8.

Step 1 (during the slot 2 and slot 1 of the modified TDMA frame). Reservation table of node N1 is shown in Table 2. Node N3 sends a route request packet to neighbor node N7 during the slot 3. If all nodes in ad hoc network do not move, N3 will reserve slot 1. In order to avoid the conflict with the node 1 due to the node mobility, N3 selects a free slot 2 to send the route request packet. N7 sends the route request packet

Table 2: Reservation table of node N1.

Node ID	Slots	Direct node	Direction
N1	S1	0	0
N2	**S3**	**1**	**0**
N3	**S2**	**2**	**0**

Table 3: Reservation table of node N3.

Node ID	Slots	Direct node	Direction
N1	S1	2	0
N2	**S3**	**1**	**0**
N3	**S2**	**0**	**0**
N7	**S1**	**1**	**0**

Table 4: Simulation parameters.

Name of parameter	Value of parameter
Network area	300 m × 300 m
Number of nodes	25~225
Transmission range	115 m
Bandwidth	2 Mb/s
Data packet size	512 bytes
Number of data slots	30
Number of sessions	20
Average message length	100 MB
MAX_SLOT_RES_TIME	10980 ms
MAX_SLOT_ALLOC_TIME	1350 ms
MAX_B	4 slots

to neighbor node N8 during the free slot 1. We can allocate the same slot to N7 with the three-hop neighbor node N1 in Table 3.

Step 2 (during the slot 3 and slot 4 of the first modified TDMA frame). Node N8 sends a route response packet to neighbor node N7 during the slot 3. N7 forwards the route response packet to neighbor node N3 during the free slot 4. The bold font in reservation tables of nodes N1 and N3 shows that the added information is after receiving the response packet and update packet.

5. Simulation

In order to verify and analyze the performance of our scheme, simulation experiments have been conducted. Traditional slot reservation approaches adopt TDMA frame as MAC layer, but our slot allocation and reservation approach is based on the modified TDMA frame. TDMA protocol class in NS 2 is modified as removing precursor MAC protocol class in our simulation. In the TDMA class declaration, remove the *tdma_preamble_statement. The physical interface for each node binds a directional antenna and adds support for multiple interfaces by using TENS [28]. We compare our scheme with a traditional slot reservation approach that uses directional antenna [18], which is called traditional scheme. We adopt (2) in Section 3.2 to compute the SNR of nodes. We

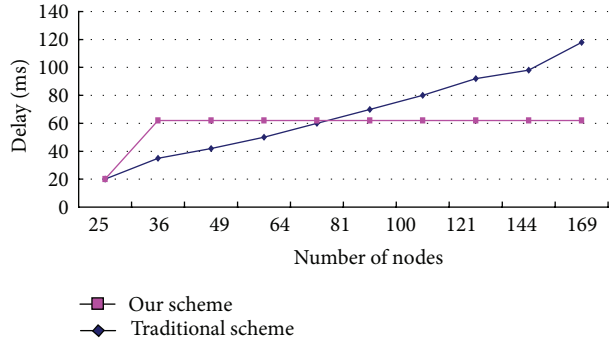


Figure 7: The maximum delay of two schemes.

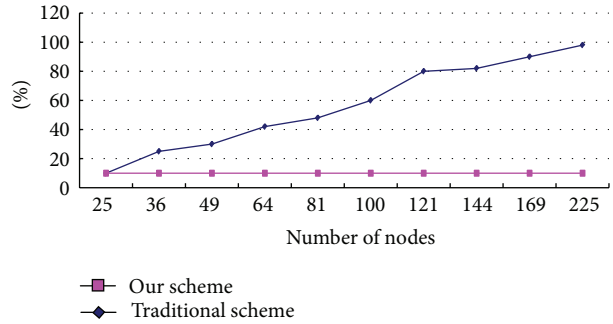


Figure 8: Percentage of the control package in the frame.

use a simulation program focusing on three performances: the maximum end-to-end delay, the percentage of control packets, and the percentage of data packets received successfully.

Simulation parameters are shown in Table 4. The nodes are allocated in an area of 300 m × 300 m. Number of slots required for each session is a random number with a uniform distribution in the range from 1 to 4 slots (1 to max_b), where max_b means the max bandwidth requirement. The transmission range of each node is 115 m. The number of data slots in a frame is 30 and the number of sessions is 20. The first performance is the maximum end-to-end delays in the busiest network case, in which all the nodes have reserved slots in the modified TDMA frame to transmit real-time data packets.

Figure 7 shows the maximum end-to-end delay value when increasing the number of nodes in network. We fix the beamwidth to 36 degrees, and the transmit power is 20 dBm. We notice that the maximum end-to-end delay using our scheme is stable and it stays 62 ms when the number of nodes is above 36 (6 × 6). However, the delay of the traditional slot reservation approach continues to increase when the number of nodes increases. The reason is that the control packets in our scheme are incorporated in the data packets. More time slots can be reserved in our scheme, and the end-to-end delay will not increase obviously. We notice also that our scheme offers better delay in dense networks (more than 100 nodes).

The second performance compared is the percentage of the control package length of the frame. Figure 8 shows

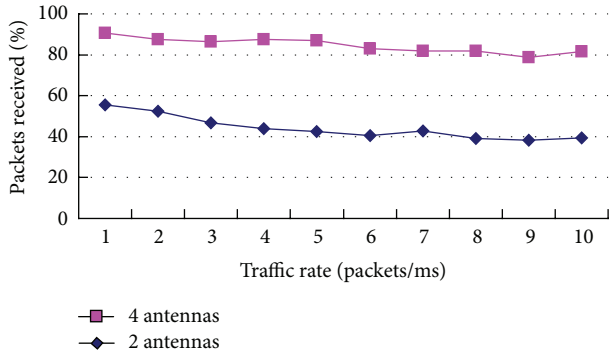


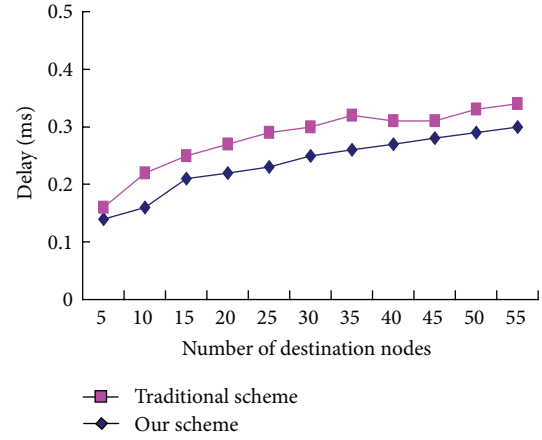
FIGURE 9: Percentage of packets received successfully.

the comparison between our scheme and the traditional slot reservation scheme. With the increase of number of nodes, the traditional scheme has more control packets. The reason is that each node in the network is allocated a control slot for transmitting control packets. Our scheme has a lower percentage of control packets of the frame, which is stable and lower than the traditional scheme. This is due to the combination of the control packets with the data packets. Control packets are not transmitted in control slots in our scheme.

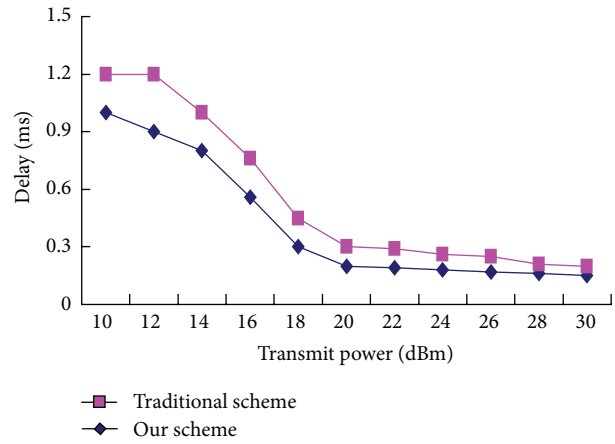
Several performance measures have been computed when the traffic rate is varied in Figure 9. The following simulations are done for two different cases: (1) 2 antennas (angle of overage is 180 degrees) and (2) 4 antennas (angle of overage is 90 degrees). The measured parameter is the overall percentage of packets received successfully. In Figure 9, it can be observed that the average overall percentage of successfully received data packets decreases when the traffic rate increases. The overall percentage of packets in four antennas case is higher than that of the two antennas case. The simulation shows that more packets can be transmitted successfully by using more directional antennas.

In following simulations, we use formula (4) of Section 3 to compute the SNR of nodes. In the fourth example, we suppose that n destinations are uniformly distributed in Figure 10(a). We fix the beamwidth to 36 degrees, and the transmit power is 20 dBm. It can be observed that with the increase of the number of destination nodes, our scheme outperforms the traditional scheme, with a delay reduction of about 18%. Figure 10(b) shows the comparison of two schemes with increasing transmit power. We suppose that the beamwidth is 36 degrees and the number of destination nodes is 30. It can be seen that our scheme always performs the traditional scheme with about 40% when transmit power is 12 dBm.

In Figure 11, we compare the overall percentage of packets received successfully when the bandwidth requirement is varied. It can be observed that with the increase of bandwidth requirement, the overall percentage of packets received successfully decreases when the beamwidth is 36 degrees and the transmit power is 20 dBm. Our scheme has higher percentage of packets received because more route request packets can be received.



(a) Delay of different destination node numbers



(b) Delay of different transmit powers

FIGURE 10: Simulation delay result for scenario of uniformly distributed.

In the final experiment, we compare the result of our scheme and the method used in [12] called DP-RQP, which uses directional antennas. Figure 12 shows the result of comparison of the two methods with increasing the bandwidth requirement. The beamwidth is 36 degree and the transmit power is 20 dBm. The number of destination nodes is 30. Our scheme has lower delay than DP-RQP method, because our scheme adopts three phases to decrease the transmission delay.

6. Conclusion

In this paper, we propose a novel time slot reservation scheme in modified TDMA-based Ad Hoc networks using directional antennas. Our scheme is based on reducing the bandwidth consumption of control packets by deleting the control phase of the traditional TDMA frame. The control packets are incorporated with the data packets. It also takes advantage of the significant increase in spatial reuse provided by using directional antennas. We propose a three-phase method to decrease the delay. Simulation results show that

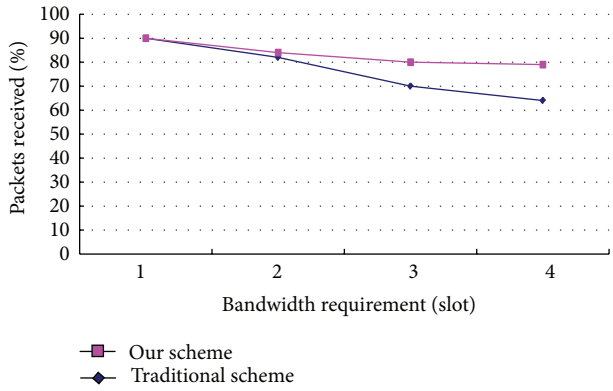


Figure 11: Overall percentage of packets received successfully.

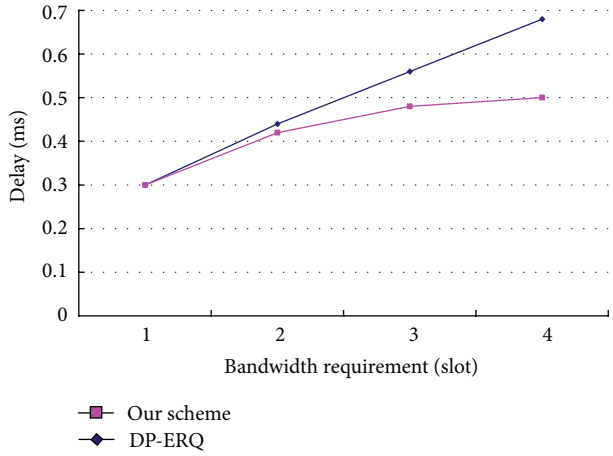


Figure 12: Simulation delay result for different bandwidth requirements.

the time slot reservation scheme using directional antennas can improve the delay and the percentage of control packets. It also increases the percentage of successfully received packets, especially when the number of directional antennas increases. Future work includes analyzing the effect of slot reservation on routing protocol and optimizing the performance of slot reservation scheme.

Acknowledgments

This work has been supported by the National Natural Science Foundation of China under no. 60772088, Young and Middle-aged Elitists' Scientific and Technological Innovation Team Project of the Institutions of Higher Education in Hubei Province (no. T200902), Yong Reserch Project of Hubei Education Department (no. Q20112205), and Key Reserch Project of Hubei Education Department (D20121903, D20121904).

References

[1] L. Chen and W. B. Heinzelman, "QoS-aware routing based on bandwidth estimation for mobile ad hoc networks," *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 3, pp. 561–572, 2005.

[2] S. M. Kamruzzaman, E. Kim, and D. G. Jeong, "An energy efficient QoS routing protocol for cognitive radio ad hoc networks," in *Proceedings of the 13th International Conference on Advanced Communication Technology (ICACT '11)*, pp. 344–349, February 2011.

[3] W. Almobaideen, K. Hushaidan, A. Sleit, and M. Qatawneh, "A Cluster-based approach for supporting QoS in Mobile Ad Hoc networks," *International Journal of Digital Content Technology and Its Applications*, vol. 5, no. 1, pp. 1–9, 2011.

[4] Y. Li, B. Sun, and X. Luo, "Joint topology-transparent scheduling and multipath QoS routing in MANETs," in *Proceedings of the 8th IEEE International Symposium on Dependable, Autonomic and Secure Computing (DASC '09)*, pp. 767–771, Chengdu, China, December 2009.

[5] N. Wang and C. Lee, "A time slot assignment scheme for multipath QoS multicast routing in mobile ad hoc networks," in *Proceedings of the International Computer Symposium (ICS '10)*, pp. 529–534, Tainan, Taiwan, December 2010.

[6] B. Sun, Y. Song, C. Gui, and Y. Jia, "Mobility entropy-based clusterhead selection algorithm for Ad Hoc networks," *Journal of Convergence Information Technology*, vol. 6, no. 10, pp. 49–55, 2011.

[7] F. W. Karam and T. Jensen, "A survey on QoS in next generation networks," *Journal of Advances in Information Sciences and Service Sciences*, vol. 2, no. 4, pp. 91–102, 2010.

[8] I. Jawhar and J. Wu, "Resource scheduling in wireless networks using directional antennas," *IEEE Transactions on Parallel and Distributed Systems*, vol. 21, no. 9, pp. 1240–1253, 2010.

[9] S. Kouroshnezhad, S. Zokaei, and H. Yeganeh, "Slot reservation demand assignment multiple access control protocol for signalling of telephony traffic via geo satellite," in *Proceedings of the 10th International Conference on Advanced Communication Technology (ICACT '08)*, pp. 693–698, February 2008.

[10] S. Sen, J. Xiong, and R. R. Choudhury, "Link layer multicasting with smart antennas: no client left behind," in *Proceedings of the 16th IEEE International Conference on Network Protocols (ICNP '08)*, pp. 53–62, October 2008.

[11] K. Sundaresan, K. Ramachandran, and S. Rangarajan, "Optimal beam scheduling for multicasting in wireless networks," in *Proceedings of the 15th Annual ACM International Conference on Mobile Computing and Networking (MobiCom '09)*, pp. 205–216, September 2009.

[12] H. Zhang, Y. Jiang, K. Sundaresan, S. Rangarajan, and B. Zhao, "Wireless data multicasting with switched beamforming antennas," in *Proceedings of the IEEE INFOCOM Mini-Conference*, pp. 526–530, Shanghai, China, April 2011.

[13] Y. Chang, Q. Liu, B. Zhang, X. Jia, and L. Xie, "Minimum delay broadcast scheduling for wireless networks with directional antennas," in *Proceedings of the 54th Annual IEEE Global Telecommunications Conference (GLOBECOM '11)*, December 2011.

[14] Y. Chang, Q. Liu, X. Jia, and K. Zhou, "Routing and transmission scheduling for minimizing broadcast delay in multi-rate wireless mesh networks using directional antennas," in *Wireless Communications and Mobile Computing*, 2012.

[15] C. Hsu, J. Sheu, and S. Tung, "An on-demand bandwidth reservation QoS routing protocol for mobile ad hoc networks," in *Proceedings of the IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing*, pp. 198–207, June 2006.

[16] X. Du, "Efficient quality-of-service provisioning and communications in hybrid wireless networks," in *Proceedings of the 7th International Conference on Wireless Communications, Networking and Mobile Computing*, keynote talk, Istanbul, Turkey, September 2011.

[17] O. Bazan and M. Jaseemuddin, "Routing and admission control for wireless mesh networks with directional antennas," in *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC '09)*, pp. 1–6, April 2009.

[18] Y. S. Chen, C. S. Hsu, and S. J. Jan, "A shoelace-based QoS routing protocol for mobile ad hoc networks using directional antenna," *Wireless Personal Communications*, vol. 54, no. 2, pp. 361–384, 2010.

[19] K. Chin, S. Soh, and C. Meng, "A novel spatial TDMA scheduler for concurrent transmit/receive wireless mesh networks," in *Proceedings of the 24th IEEE International Conference on Advanced Information Networking and Applications (AINA '10)*, pp. 481–488, April 2010.

[20] Y. Li and X. Luo, "A bandwidth allocation scheme based on QoS in satellite networks using directional antennas," in *Proceedings of the 7th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM '11)*, pp. 1–4, September 2011.

[21] P. Feng, Y. Ding, B. Liu, J. Wu, L. Gui, and H. Zhou, "A QoS constrained cognitive routing algorithm for ad hoc networks based on directional antenna," in *Proceedings of the International Conference on Wireless Communications and Signal Processing (WCSP '11)*, November 2011.

[22] Q. Liu, X. Jia, and Y. Zhou, "Topology control for multi-channel multi-radio wireless mesh networks using directional antennas," *Wireless Networks*, vol. 17, no. 1, pp. 41–51, 2011.

[23] N. D. Sidiropoulos, T. N. Davidson, and Z. Luo, "Transmit beamforming for physical-layer multicasting," *IEEE Transactions on Signal Processing*, vol. 54, no. 6 I, pp. 2239–2251, 2006.

[24] A. Lozano, "Long-term transmit beamforming for wlreless multicasting," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (IICASSP '07)*, pp. III417–III420, April 2007.

[25] E. Matskani, N. D. Sidiropoulos, and L. Tassiulas, "On multicast beamforming and admission control for UMTS-LTE," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (IICASSP '08)*, pp. 2361–2364, April 2008.

[26] F. B. Cross, *Smart Antennas for Wireless Communications*, McGraw-Hill, New York, NY, USA, 2005.

[27] U. Feige, "A threshold of In n for approximating set cover," *Journal of the ACM*, vol. 45, no. 4, pp. 634–652, 1998.

[28] The Enhanced Network Simulator (Version 1.2), http://www.cse.iitk.ac.in/users/braman/tens/.

Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2013, Article ID 160523, 8 pages
http://dx.doi.org/10.1155/2013/160523

Research Article

Distributed Fault-Tolerant Event Region Detection of Wireless Sensor Networks

Dyi-Rong Duh,[1] Ssu-Pei Li,[2] and Victor W. Cheng[2]

[1] *Department of Computer Science and Information Engineering, Hwa Hsia Institute of Technology, New Taipei City 23568, Taiwan*
[2] *Department of Computer Science and Information Engineering, National Chi Nan University, Nantou Hsien 54561, Taiwan*

Correspondence should be addressed to Dyi-Rong Duh; drduh@cc.hwh.edu.tw

Received 23 January 2013; Accepted 10 August 2013

Academic Editor: Hongju Cheng

This work provides a distributed fault-tolerant event region detection algorithm for wireless sensor networks. The proposed algorithm can identify faulty and fault-free sensors and ignore the abnormal readings to avoid false alarm. Moreover, every event region can also be detected and identified. Simulation results show that fault detection accuracy (FDA) is greater than 92%, false alarm rate (FAR) is near 0%, and event detection accuracy (EDA) is greater than 99% under uniform distribution. FDA is greater than 92%, FAR is less than 1.2%, and EDA is greater than 88% under random distribution when sensor fault probability is less than 0.3.

1. Introduction

The wireless sensor network (WSN) is a novel technology developed in recently years [1–9]. A wireless sensor network (WSN) consists of a large number of sensors. These sensors are used to monitor environmental variations such as temperature, humidity, pressure, and concentration of chemicals. Sensors in a WSN have limited computation, communication, and sensing capabilities because they are low cost and energy-constrained. Moreover, sensors are often deployed in an uncontrolled and harsh environment. They are prone to be faulty and hard to maintain. Therefore, a fault-tolerant and energy-efficient algorithm is required for network operation [4, 10–13].

The communication protocol of a WSN is very important in order to reduce power consumption. Three general-purpose protocol architectures include direct transmission protocol [15, 16], routing protocol [17–19], and clustering protocol [1, 3, 7, 11, 20–24]. Clustering is a popular energy-efficient protocol in WSNs [3, 11, 21, 23]. It divides the monitored area into several clusters, and each cluster has a cluster head. A cluster head can be regarded as a local fusion center. Each cluster member will send its physical value or decision to its cluster head, and the cluster head can thus make data aggregation or data fusion to eliminate invalid values and report this result to the base station (BS). Significantly, the challenge of designing a clustering WSN is that a cluster head is easy to exhaust its energy, and hence the network will be out of control. Heinzelman et al. proposed the LEACH algorithm [11]. In LEACH, cluster heads are randomly chosen, and a self-organization procedure is performed. This protocol balances the energy load between sensors. Heinzelman et al. later proposed the improved centralized algorithm as LEACH-C [21]. In LEACH-C, each sensor takes its turn as the cluster head according to the remaining energy. Therefore, the energy load is more evenly balanced than that in the LEACH algorithm. Nocetti et al. proposed a clustering algorithm based on connectivity [23]. The efficiency of this algorithm can be measured by the number of clusters formed and the number of border nodes produced.

Chen et al. proposed a distributed fault detection algorithm in [25]. Each sensor compares the observed data with its neighbors and makes a local decision by majority of votes. Krishnamachari and Iyengar proposed a distributed Bayesian algorithm [26]. In this algorithm, faults can be detected and corrected, but it also introduces new errors. Wu et al. [14] proposed an algorithm to solve the fault-event disambiguation problem. They focused on two typical cases, the event regions with ellipses or straight lines as boundaries.

This work provides a distributed and fault-tolerant algorithm to extend the uses of fault-event disambiguation. It can identify not only faulty and fault-free sensors but

also the region where the event occurs. Simulation results demonstrate that the proposed algorithm has high accuracy and low false alarm in any shape of event regions.

The rest of this paper is organized as follows. Section 2 defines the network model and fault model. Section 3 describes the algorithm. Section 4 makes some simulations and shows their results. Conclusion and future work are drawn in Section 5.

2. Network Model and Fault Model

For ease of reference, frequently used notations and definitions in this paper are listed in Table 1. Each sensor s_i in a WSN has an observation x_i from its location. This observation is independent and identically distributed on phenomenon from sensor to sensor. θ_d is defined as a threshold for checking whether two readings are in the same situation. For example, to get the relationship and to decide the status of two sensors s_i and s_j, the measurement difference d_{ij} is compared with θ_d. When $d_{ij} > \theta_d$, at least one of s_i and s_j is abnormal. θ_{ev} is defined as the event threshold. When s_i is a fault-free sensor and x_i is greater than θ_{ev}, s_i is identified as an event sensor. θ_D and θ_f are defined as the thresholds of degree and faith, respectively. These two thresholds are used in the procedure of making the strong decision by a cluster head, which will be further described in the next section. Assume that each sensor has a unique ID and can use the power control mechanism to vary the transmitting power, and the transmission range in the WSN is independent from sensor to sensor. If a sensor is far away from all other sensors, it can enhance its transmitting power to increase the transmission range and communicate with other sensors.

Notably, the fault model of this work assumes that computation and communication capabilities of each sensor always work properly. Three types of sensing fault are defined as stuck-at-maximum fault, stuck-at-minimum fault, and random fault. When a sensor is on the stuck-at-maximum fault, on the stuck-at-minimum fault, or on the random fault, it reports the maximum physical value, the minimum physical value, or a random physical value uncorrelated to the environment, respectively. These faulty sensors make the network unreliable and affect the final decision of the WSN. Therefore, a fault-tolerant algorithm to identify and isolate faulty sensors is designed in this work, and hence the precision of the event region detection is improved.

Table 1: Notations and definitions.

n	Total number of sensors
s_i	The sensor with ID i
$N(s_i)$	The set of neighbors of s_i
$M(s_i)$	The set of neighbors of s_i that have not joined any cluster
D_i	The degree of s_i
B_i	The set of cluster(s) that s_i belongs to
x_i	The measurement of s_i
d_{ij}	The measurement difference of s_i and s_j, $d_{ij} = \lvert x_i - x_j \rvert$
c_{ij}	The measurement status between s_i and s_j, $c_{ij} \in \{0, 1\}$, $c_{ij} = c_{ji}$
w_i	The weight of cluster head i
f_i	The faith of cluster head i, $f_i = w_i/D_i$
P_i	The status of s_i, $P_i \in$ {GD, FT, EV, UD}
θ_d	The measurement difference threshold
θ_D	The degree threshold
θ_f	The faith threshold
θ_{ev}	The predefined event threshold
T_{rcv}	The timer for receiving cluster invitation
T_{self}	The timer for the self-decision
T_{weak}	The timer for the weak decision

3. Proposed Algorithm

Assumes that the BS is capable of broadcasting a global message to sensors everywhere and each sensor knows its own geographical location either through GPS or RF-based beacons [27]. Each cluster head collects data from its cluster members. The data may contain abnormal readings. Therefore, this work provides an algorithm to identify those sensors that get abnormal readings. The proposed algorithm adopts the correlative relationship to distinguish event sensors or faulty sensors. Faulty sensors are likely to be uncorrelated, and sensors in the event region are spatially correlated. The algorithm is divided into two phases, clustering phase and decision phase.

Some other definitions are described as follows. If $d_{ij} \leq \theta_d$, where s_i is a cluster head and $s_j \in N(s_i)$, then $c_{ij} = c_{ji} = 0$ and w_i increases by one. If $d_{ij} > \theta_d$, then $c_{ij} = c_{ji} = 1$ and w_i does not increase. Weight w_i is used to calculate f_i. Four statuses of a sensor in a WSN are defined as Good (GD), Faulty (FT), Event (EV), or Undetermined (UD).

3.1. Clustering Phase. The clustering phase starts when the BS broadcasts a clustering message to all sensors. In this phase, the monitored environment is divided into several clusters by cluster heads. Each cluster head can be regarded as a local fusion center. It collects information from its members and reports the fusion to the remote BS. Therefore, a well-designed clustering rule is to avoid chaotic situations such that a cluster head has no members or two cluster heads exist in the same cluster. The clustering rule of this work uses the degree of a sensor as the primary key and the sensor ID as the secondary key. At the first stage, each sensor exchanges degrees with its neighbors. Every sensor s_j, which has $D_j = 0$ and does not belong to any cluster, is classified as an isolated sensor. Every sensor s_i with the highest degree among $M(s_i)$ and itself is chosen as the cluster head. The remaining sensors start a timer T_{rcv} for receiving invitations. If two or more sensors have the same maximum degree as s_i and $M(s_i)$, the sensor with the lowest sensor ID is chosen to be the cluster head. Each cluster head sends an invitation to its neighbors to include them as its cluster members and ends the clustering phase. When a sensor receives an invitation, it joins the cluster. Every sensor will check if it has joined any cluster

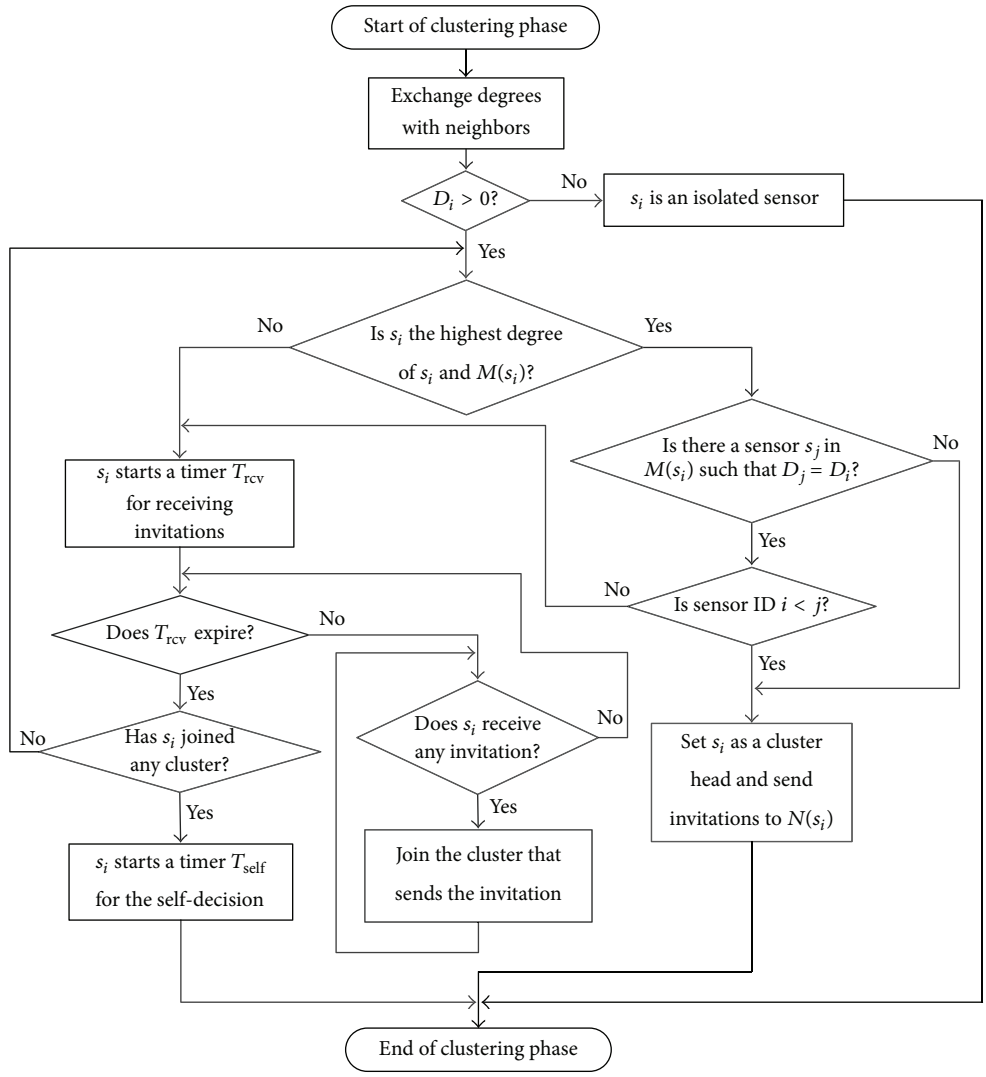


Figure 1: The flow chart of the clustering phase.

when T_{rcv} expires. If yes, it starts a timer T_{self} for the self-decision which will be described in the decision phase, and the clustering phase is completed. If not, it continues to check if it is the highest degree among $M(s_i)$ and itself as is in the previous step. In the clustering phase of the algorithm, each cluster head forms its own cluster and each member sensor may belong to more than one cluster. Because all neighbors of a cluster head are its cluster members, the clustering rule used in the algorithm guarantees that every cluster head is not adjacent to any other cluster head; that is, there is only one cluster head in each cluster. Figure 1 shows the flow chart of the clustering phase.

3.2. Decision Phase. Initially, the status of each sensor is set to UD. This phase is divided into three kinds of decision: strong decision, weak decision, and self-decision. An important mechanism in this phase is that when sensor s_j is set to GD or EV, it will cease to allow other sensors to reset its status. On the other hand, when s_j is set to FT, P_j may be switched to the same status as P_k by s_k if $s_k \in N(s_j)$ and P_k = GD or EV to reduce false alarms. The strong decision is only performed in the cluster heads whose degree and faith exceed the thresholds. The weak decision is performed in the cluster heads with status UD when T_{weak} expires. The self-decision is performed in the cluster members with status UD when T_{self} expires. Each decision is described in detail as follows.

3.3. Strong Decision. First, each member sensor s_j sends the observation x_j to its cluster head(s), and each cluster head s_i calculates d_{ij}, c_{ij}, w_i, D_i, and f_i where $s_j \in N(s_i)$. If $D_i \geq \theta_D$, $f_i \geq \theta_f$, and P_i = UD, cluster head s_i sets P_i to GD when $x_i < \theta_{ev}$ or EV when $x_i \geq \theta_{ev}$ and determines the statuses of $N(s_i)$. If $D_i \geq \theta_D$, $x_i \geq \theta_{ev}$, $f_i \geq \theta_f/2$, and P_i = UD, cluster head s_i sets P_i to EV and also determines the statuses of $N(s_i)$. Otherwise, s_i starts a timer T_{weak} for the weak decision described in next paragraph and exits current decision. Second, during the status propagation, s_i checks c_{ij} to set P_j to UD or FT for all $s_j \in N(s_i)$. If $c_{ij} = 0$, then P_j is copied from P_i, and s_j continues to set the statuses of $N(s_j)$. If $c_{ij} = 1$ and P_i = GD, P_j is set to FT, and s_j terminates

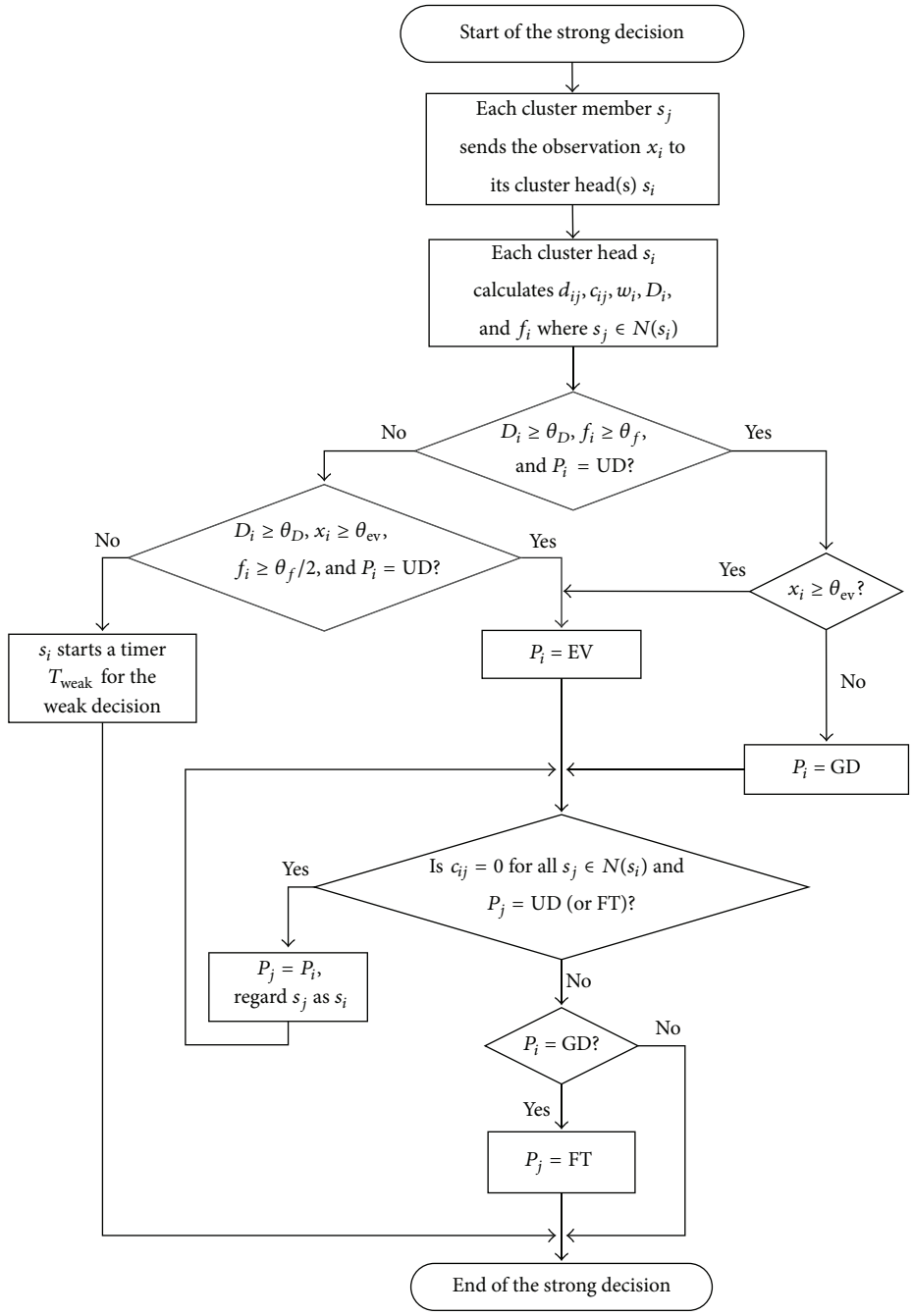


Figure 2: The flow chart of the strong decision.

the status propagation. Figure 2 illustrates the flow chart of the strong decision.

3.4. Weak Decision. When T_{weak} of cluster head s_i expires, s_i with P_i = UD starts the weak decision. Based on majority vote, if $f_i > 0.5$, s_i sets P_i to GD when $x_i < \theta_{ev}$ or EV when $x_i \geq \theta_{ev}$, and s_i checks c_{ij} to set P_j to UD or FT for all $s_j \in N(s_i)$. If $c_{ij} = 0$, P_j is copied from P_i. Moreover, if $c_{ij} = 1$ and P_i = GD, P_j is set to FT. If $f_i \leq 0.5$, s_i sets P_i to FT and terminates the weak decision. Figure 3 illustrates the flow chart of the weak decision.

3.5. Self-Decision. When T_{self} of member sensor s_j expires, s_j first starts reclustering described in next paragraph and then checks P_j. If P_j = UD, s_j searches for a GD sensor s_k in $N(s_j)$ and calculates c_{jk}. If $c_{jk} = 0$, P_j is set to GD. If $c_{jk} = 1$, P_j is set to FT. Notably, it is possible that no sensor with status GD can be found in $N(s_j)$. When that happens, s_j uses its

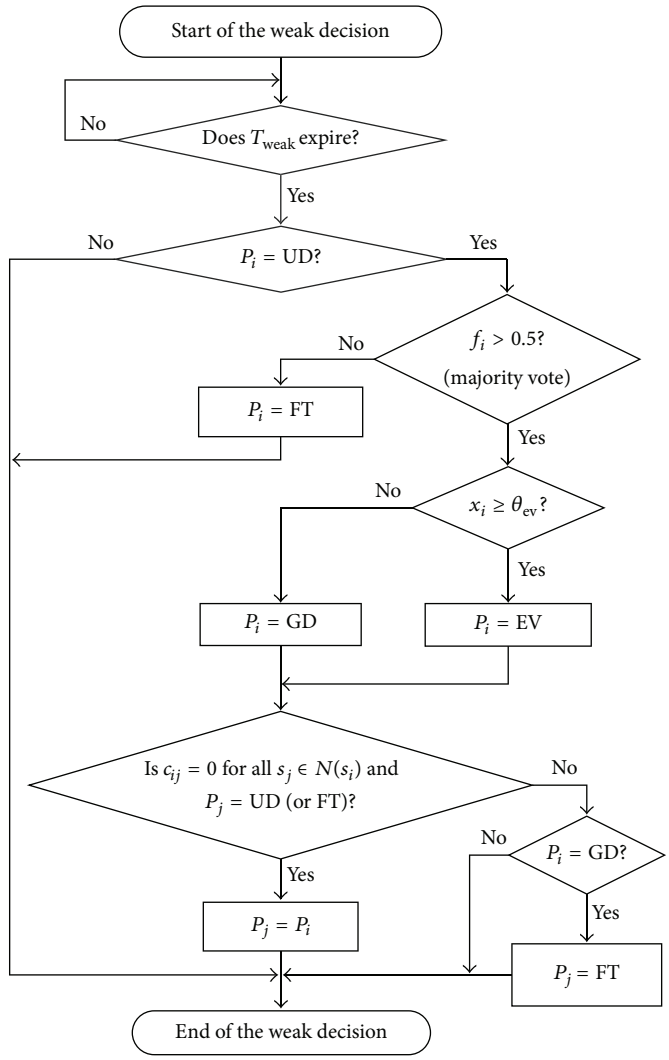


Figure 3: The flow chart of the weak decision.

maximum transmission range for searching a sensor with status GD. Nevertheless, if a sensor with status GD cannot be found within the range, P_j is set to FT. Figure 4 illustrates the flow chart of the self-decision.

Reclustering is an important approach in the self-decision. If a cluster head s_i is set to FT, its members who do not belong to any other cluster must be reclustered because a faulty cluster head cannot properly report data to the BS. The erroneous data may cause incorrect final decisions at the BS and thus the WSN may miss some event regions. For example, s_i and s_h are two cluster heads, and P_i is set to FT, and P_h is set to GD. There exist two cluster members s_j and s_k of cluster s_i such that $B_j = \{i, h\}$ and $B_k = \{i\}$. In this case, since s_i is faulty, s_j removes i from B_j and belongs to cluster h only. Therefore, s_j does not require reclustering. On the other hand, after removing i from B_k, s_k does not belong to any cluster and its observations will be ignored. Therefore, s_k requires reclustering. After reclustering, everything is accomplished by the new cluster head. Every sensor is included in some cluster after the decision phase, except for all isolated sensors.

4. Simulation Results

The simulation platform is a PC with Windows XPSP3, and the simulation program is developed in C#. The network parameters in this simulation are preset as follow. The transmission range of a sensor is between 2 m and 10 m. Thresholds θ_d, θ_{ev}, and θ_f are set to 5, 60, and 0.66, respectively, and θ_D is set to the average degree of the network. Notably, with the aid of experiments, we tuned every threshold carefully to get higher FDA and EDA and lower FAR. Each physical datum of any observation is between 0 and 100. The fault detection accuracy (FDA) is defined as the ratio of the number of faulty sensors detected to the total number of faulty sensors. The false alarm rate (FAR) is defined as the ratio of the number of fault-free sensors diagnosed as faulty to the total number of fault-free sensors. The event detection accuracy (EDA) is defined as the ratio of the number of event sensors detected to the total number of sensors located in the event region.

The result of the proposed algorithm will be compared with that of a recent algorithm for event boundary detection

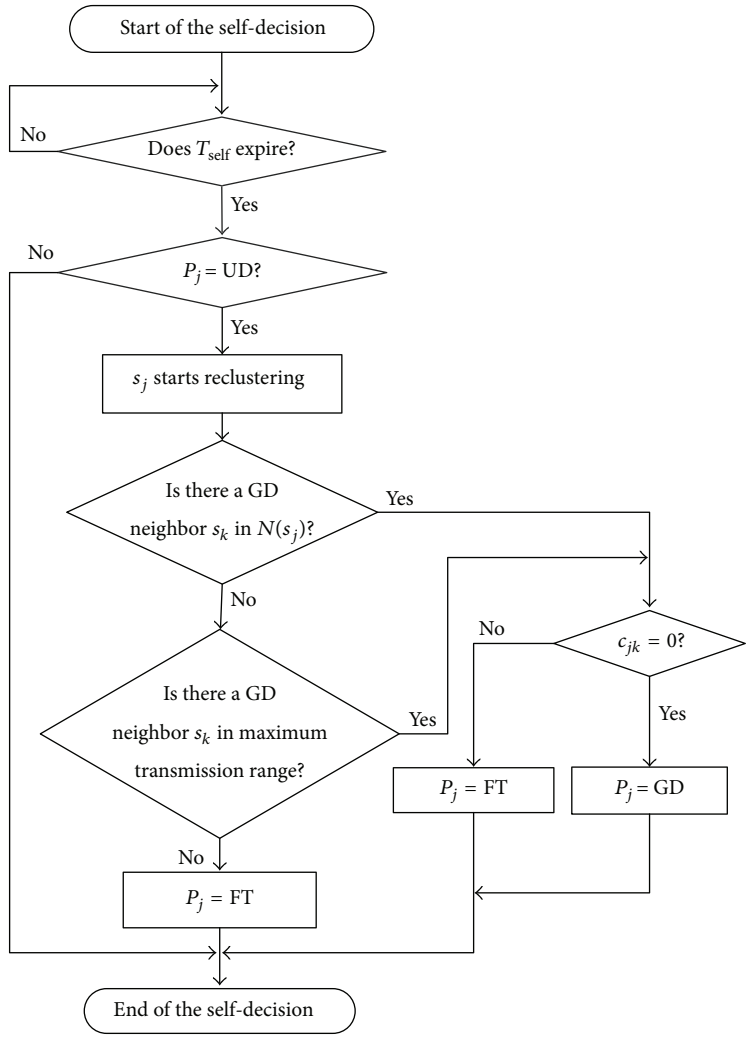


Figure 4: The flow chart of the self-decision.

with faulty sensors in a WSN by Wu et al. [14]. For ease of comparison, the network setup is the same as that of Wu et al. 4,096 sensors are uniformly distributed in a 64 m × 64 m square region and represent an averaged summary of 100 runs. Figure 5 illustrates the FDA of the algorithm provided by Wu et al. and the proposed algorithm. Figure 6 shows the FAR of Wu et al.'s algorithm and the proposed algorithm.

Referring to Figure 5, the FDA of Wu et al.'s algorithm is slightly better than that of the proposed algorithm when average degree is between 20 and 50. This is because our simulation includes random faults, and random faulty sensors may catch random physical data near normal range (or event range), which are difficult to be correctly detected. Nevertheless, the FDA of Wu et al.'s algorithm is very inaccurate, while average degree is 10. In contrast, the FDA of the proposed algorithm remains close to 93% under various average degrees. The FARs of Wu et al.'s algorithm and the FAR of the proposed algorithm are illustrated in Figure 6. Undoubtedly, higher FAR always gets higher FDA. Obviously, our algorithm thus makes a significant improvement in FAR since it is almost 0%. The key point is that when a fault-free sensor s_j has been diagnosed as FT, its status may be switched to GD as long as there exists a sensor with status GD in $N(s_j)$ running the status propagation.

Figure 7 demonstrates that the FDA of our algorithm decreases smoothly with the sensor fault probability from 0.1 to 0.4 and average degree 10. Actually, a high degree network demands high cost and is not practical for a large scale network. Therefore, the proposed algorithm is more suitable for WSNs.

Figure 8 shows the results of our algorithm that 250 sensors are randomly distributed in a 30 m × 30 m square region with average degree of 10, and the predefined event region is a square with side length 8 m located in the middle. The FDA is greater than 92%, the FAR is less than 1.2%, and EDA is greater than 88% when sensor fault probability is less than 0.3. Figure 9 illustrates the simulation result that the BS uses the convex hull algorithm to identify the event region. In Figure 9, red, black, and blue nodes indicate GD, FT, and EV sensors, respectively. The black circles are clusters, the green square is the predefined event region, and the purple polygon is the detected event region.

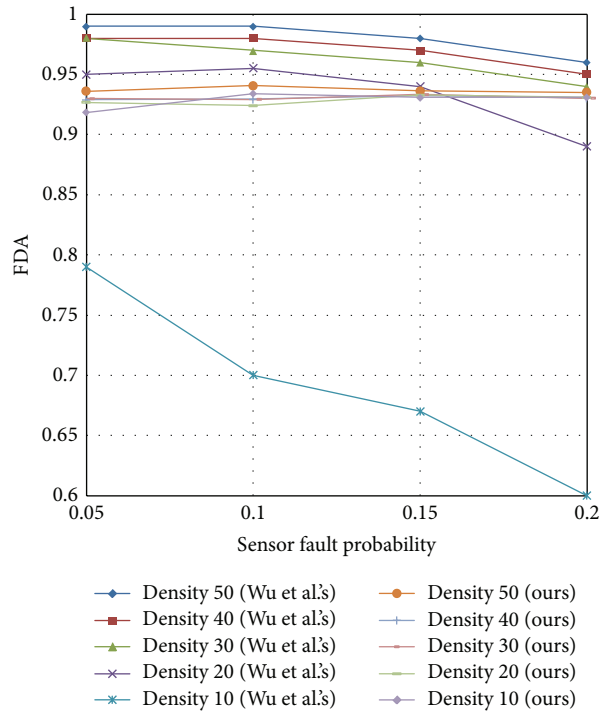


Figure 5: The FDA of Wu et al.'s algorithm [14] and the proposed algorithm.

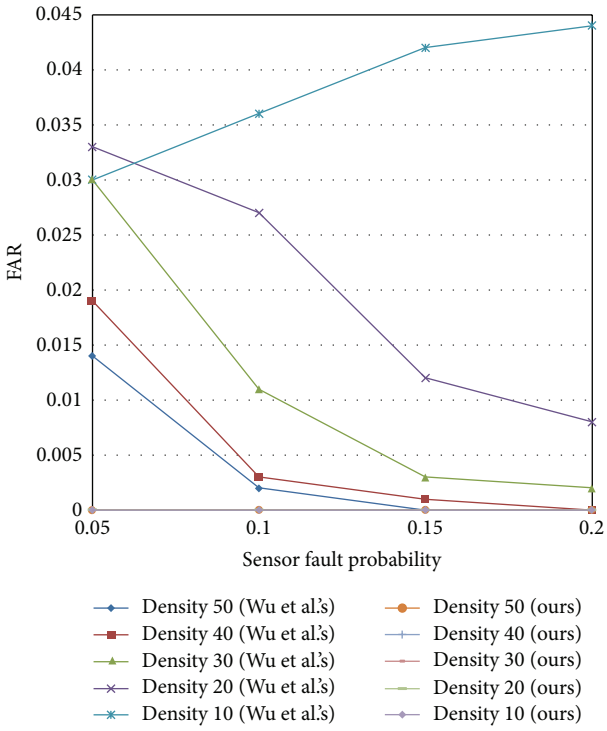


Figure 6: The FAR of Wu et al.'s algorithm [14] and the proposed algorithm.

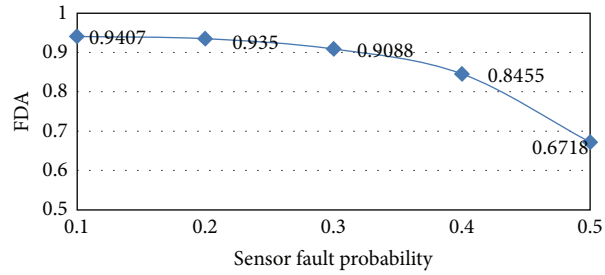


Figure 7: The FDA of our algorithm in a WSN with an average degree of 10.

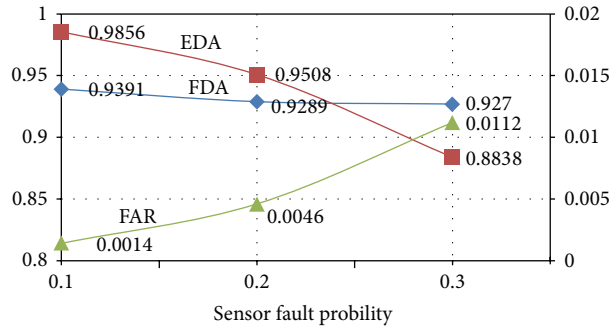


Figure 8: FDA, FAR, and EDA of our algorithm in a WSN when the sensors are randomly distributed with an average degree of 10.

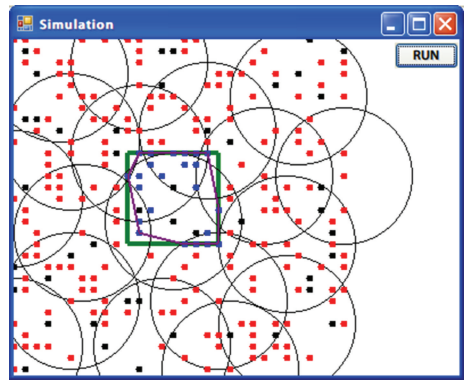


Figure 9: Convex hull algorithm is used to identify the event region at the BS. (Red: GD sensor; black: FT sensor; blue: EV sensor; green: the predefined event region; purple: the detected event region).

5. Conclusion

This work provides a distributed fault-tolerant algorithm for event region detection of WSNs to solve the fault-event disambiguation problem. It may be widely used in abnormal region detection. For example, sensors can be deployed in a forest by an airplane to discover promptly forest fires by monitoring the temperature around each sensor. The BS can identify the abnormal region based on the report of the cluster heads and send out a fire alarm signal. The proposed algorithm has high accuracy on event detection and low false alarm. Our future work will focus on power consumption and cluster head rotation to extend system life time. Moreover,

to compute the best value of every threshold is our further research.

Acknowledgment

The authors would like to thank the National Science Council, Taiwan, for financially supporting this research under Contract NSC 100-2221-E-146-014.

References

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

[2] J. Agre and L. Clare, "An integrated architecture for cooperative sensing networks," *IEEE Computer*, vol. 33, no. 5, pp. 106–108.

[3] N. Amini, A. Vahdatpour, W. Xu, M. Gerla, and M. Sarrafzadeh, "Cluster size optimization in sensor networks with decentralized cluster-based protocols," *Computer Communications*, vol. 35, no. 2, pp. 207–220, 2012.

[4] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Communications Magazine*, vol. 40, no. 8, pp. 102–114, 2002.

[5] D. Culler, D. Estrin, and M. Srivastava, "Overview of sensor networks," *Computer*, vol. 37, no. 8, pp. 41–49, 2004.

[6] D. D. Geeta, N. Nalini, and R. C. Biradar, "Fault tolerance in wireless sensor network using hand-off and dynamic power adjustment approach," *Journal of Network and Computer Applications*, vol. 36, no. 4, pp. 1174–1185, 2013.

[7] J.-W. Lin and D.-R. Duh, "Distributed fault detection of wireless sensor networks," in *Proceedings of the 2010 International Conference on Wireless Networks (ICWN '10)*, pp. 472–477, Monte Carlo Resort, Las Vegas, Nev, USA, July 2010.

[8] J.-Y. Wu, D.-R. Duh, and T.-Y. Wang, "On-line fault-tolerant decision fusion scheme based on a record table in wireless sensor networks," in *Proceedings of the Conference on Information Technology and Applications in Outlying Islands (ITAOI '13)*, pp. 249–255, Jinning, Kinmen, Taiwan, May 2013.

[9] D. Xia and N. Vlajic, "Near-optimal node clustering in wireless sensor networks for environment monitoring," in *Proceedings of the International Conference on Electrical and Computer Engineering (CCECE '06)*, pp. 1825–1829, Ottawa, Ontario, Canada, May 2006.

[10] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar, "Next century challenges: scalable coordination in sensor networks," in *Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM '99)*, pp. 263–270, Seattle, Wash, USA, August 1999.

[11] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *Proceedings of the 33rd Annual Hawaii International Conference on System Siences (HICSS-33 '00)*, pp. 1–10, January 2000.

[12] A. Manjeshwar and D. P. Agrawal, "TEEN: a routing protocol for enhanced efficiency in wireless sensor networks," in *Proceedings of the 15th International Symposium on Parallel and Distributed Processing (IPDPS '01)*, vol. 3, pp. 2009–2015, San Francisco, Calif, USA, April 2001.

[13] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient MAC protocol for wireless sensor networks," in *Proceedings of the 21st Annual Joint Conference of the IEEE International Conference on Computer and Communications Societies (INFOCOM '02)*, pp. 1567–1576, New York, NY, USA, June 2002.

[14] W. Wu, X. Cheng, M. Ding, K. Xing, F. Liu, and P. Deng, "Localized outlying and boundary data detection in sensor networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 8, pp. 1145–1156, 2007.

[15] L.-C. Chen, D.-R. Duh, and T.-Y. Wang, "Using a saturation counter to improve the accuracy of final decisions in wireless sensor networks," in *Proceedings of the International Conference on Wireless Networks (ICWN '09)*, pp. 124–129, Monte Carlo Resort, Las Vegas, Nev, USA, July 2009.

[16] T.-Y. Wang, L.-Y. Chang, D.-R. Duh, and J.-Y. Wu, "Fault-tolerant decision fusion via collaborative sensor fault detection in wireless sensor networks," *IEEE Transactions on Wireless Communications*, vol. 7, no. 2, pp. 756–768, 2008.

[17] M. Ettus, "System capacity, latency, and power consumption in multihop-routed SS-CDMA wireless networks," in *Proceedings of the IEEE Radio and Wireless Conference (RAWCON '98)*, pp. 55–58, Colorado Springs, Colo, USA, August 1988.

[18] T. J. Shepard, "Channel access scheme for large dense packet radio networks," in *Proceedings of the Annual Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM '96)*, pp. 219–230, Stanford, Calif, USA, August 1996.

[19] S. Singh, M. Woo, and C. S. Raghavendra, "Power-aware routing in mobile and ad hoc networks," in *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM '98)*, pp. 181–190, Dallas, Tex, USA, 1998.

[20] D. J. Baker, A. Ephremides, and J. A. Flynn, "The design and simulation of a mobile radio network with distributed control," *IEEE Journal on Selected Areas in Communications*, vol. 2, no. 1, pp. 226–237, 1984.

[21] 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.

[22] C. R. Lin and M. Gerla, "Adaptive clustering for mobile wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 15, no. 7, pp. 1265–1275, 1997.

[23] F. G. Nocetti, J. S. Gonzalez, and I. Stojmenovic, "Connectivity-based k-hop clustering in wireless networks," *Telecommunication Systems*, vol. 22, no. 1–4, pp. 205–220, 2003.

[24] R. Ruppe, S. Griswald, P. Walsh, and R. Martin, "Near Term Digital Radio (NTDR) system," in *Proceedings of the 1997 Military Communications Conference (MILCOM '97)*, vol. 3, pp. 1282–1287, Monterey, Calif, USA, November 1997.

[25] J. Chen, S. Kher, and A. Somani, "Distributed fault detection of wireless sensor networks," in *Proceedings of the Workshop on Dependability Issues in Wireless Ad Hoc Networks and Sensor Networks (DIWANS '06)*, pp. 65–71, Los Angeles, Calif, USA, September 2006.

[26] B. Krishnamachari and S. Iyengar, "Distributed Bayesian algorithms for fault-tolerant event region detection in wireless sensor networks," *IEEE Transactions on Computers*, vol. 53, no. 3, pp. 241–250, 2004.

[27] N. Bulusu, J. Heidemann, and D. Estrin, "GPS-less low-cost outdoor localization for very small devices," *IEEE Personal Communications*, vol. 7, no. 5, pp. 28–34, 2000.

Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2013, Article ID 625070, 11 pages
http://dx.doi.org/10.1155/2013/625070

Research Article

IM-Dedup: An Image Management System Based on Deduplication Applied in DWSNs

Jilin Zhang, Shuting Han, Jian Wan, Baojin Zhu, Li Zhou, Yongjian Ren, and Wei Zhang

Department of Computer and Technology, Hangzhou Dianzi University, Hangzhou, Zhejiang 310018, China

Correspondence should be addressed to Jian Wan; wanjian@hdu.edu.cn

Received 9 February 2013; Accepted 18 March 2013

Academic Editor: Neal N. Xiong

In distributed wireless sensor networks (DWSNs), the data gathered by sink is always massive and consumes a lot of resources. It is suitable for cloud computing platform to apply service in data processing system. In cloud computing, IAAS platform provides services and calculation to the user through the virtual machine. The management of virtual machine images not only consumes a huge amount of storage space but also gives large pressure on network transmission. By using deduplication technology in openstack, this paper designed and implemented, an image management system IM-dedup, which uses static chunking (SC) to divide image file into blocks of data, avoid duplication data blocks transmission on network by using fingerprint pretransmission technology, and reduce storage space by deploying kernel mode file system with deduplication in the image storage server. The experimental results showed that the system not only reduced 80% usage of the virtual machine image storage, but also saved at least 30% of transmission time. Furthermore, the research on virtual machine image format showed that "VMWare Virtual Machine Disk Format" (VMDK), "Virtual Desktop Infrastructure" (VDI), "QEMU Copy On Write2" (QCOW2), and RAW image formats are more suitable for the IM-dedup system.

1. Introduction

Deduplication technology is a lossless data compression technology, which first utilizes the characteristics of the data to detect duplicated objects in data streams. After detection of duplications, it replaces the duplicated copies of the data object by using pointer to the unique data object. Deduplication technology is able to share duplicated data blocks across files, which leads to a high data compression rate, and therefore has been widely used in data backup archive field [1–5].

Gathering remote information is an important part of applications in distributed wireless sensor networks (DWSNs). For example, monitoring temperature, lightness, humidity, atmospheric pollution, building structure integrity, and chemical and biological threats. Since the data gathered by sinks is always massive and consumes a lot of resources, it is suitable to apply a cloud computing platform to host and process massive data in such a situation; the architecture is shown in Figure 1. IAAS cloud computing platforms, such as eucalyptus [6], EC2 [7], and openstack [8], use virtual machines to provide computing services for the users. Since the needs of users are flexible and different, it should support various virtual machines with a variety of configurations, such as different lineage operating system, or 32/64 bit system. In order to establish a virtual machine quickly, a cloud computing system uses virtual machine images to manage virtual machines. A single user may have multiple differently configured virtual machine images, each of which is usually larger than 1 GB. With the expansion of cloud scale, the cost of the management of virtual machine images in cloud computing systems increases fast, particularly the transmission time consumption and image storage consumption. The comparison of time consumption between MD5 and SHA1 is shown in Tables 1 and 2 gives the comparison of average execution time and CPU usage between MD5 and SHA-1.

Some recent studies [9, 10] have shown that more than 80% of the data blocks in virtual machine storage are duplicated. In other words, the 80% of the data blocks are shared by different versions of virtual machine image [11]. Therefore, the application of deduplication technology is helpful for IAAS cloud computing platforms to manage data

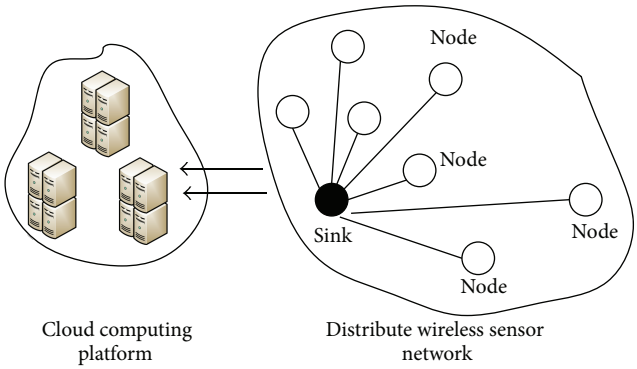


Figure 1: The architecture of data processing system in distribute wireless sensor networks.

Table 1: The comparison of time consumption between MD5 and SHA1 [27].

Hash function	1 K	100 M	1.5 G
Md5	1 ms	0.931 ms	20.762 s
SHA-1	1 ms	0.987 ms	21.518 s

Table 2: The comparison of average execution time and CPU usage between MD5 and SHA-1 [27].

Hash function	MD5	SHA-1
Average execution time	48 S	48.2 S
CPU usage	28%	72%

Test data set is vmware virtual operating system disc files, Windows 2003, CPU: P4, 28 G, AM: lG, D: 4lXBDE.

more efficiently. Most of the existing researches are concerned about how to deduplicate virtual machines, but few of them are concerned on particular open source cloud computing platforms. Some commercial file systems also can support deduplication technology, such as SDFS [12], ZFS [13], which can achieve high IO throughput. But all of them are designed for enterprise applications and require a big memory which is unsuitable for openstack.

In this paper, an image management system, named IM-dedup, is presented to solve the problems and difficulties of the openstack cloud-computing platform. The presented system applies static block technology to divide a file into blocks of data, uses the fingerprint pretransmission technology to avoid transmission of duplicated data blocks, deploys the kernel-space deduplication file system on the image storage server to reduce storage space, and uses the memory filter to reduce the overhead of disk index. Besides, it also improves the locality of data by centralizing fingerprints in disk to achieve a higher IO throughput rate with the limited memory occupancy rate.

The experiment showed that the system not only reduced at most 80% of storage capacity expenses on virtual machine image storage but also saved at least 30% of transmission time. In addition, our results on different image formats of virtual machine showed that VMDK, VDI, QCOW2, and RAW image formats are more suitable for this system.

2. Related Work

In the field of data storage and backup research, deduplication is the most important emerging technology. The complete process of deduplication consists of four steps: file chunking, fingerprint extraction, fingerprint lookup, and data storage. Researchers have made studies of deduplication technology from various perspectives, mainly focused on the following three aspects.

2.1. Studies on Strategies of Data Partitioning and Feature Extraction. AA-dedupe [14] is an application-aware deduplication technology, which applies different methods to chunk and calculate fingerprints according to file types. To improve efficiency, some excessively small files (smaller than 1 k) are ignored. SAM [15] is a semantic-aware deduplication technology. It can reduce deduplicated data blocks selectively by analyzing the feature of the file size, file location, and file type. In addition, SAM combines global-file-level deduplication with local-block-level deduplication to find the trade-off point between duplication rate and system performance.

2.2. Studies on Strategies of Memory Index Lookup. As mentioned in many papers [11, 16–18], fingerprint lookup has seriously affected IO performance in large-scale systems. Therefore, how to improve the efficiency of index query has become one of the hottest research spots in the field of deduplication technology. DDFS [11] uses Bloom filter and cache technology to improve the performance of data block lookup. Due to the use of Bloom filter, DDFS demands a big memory and thus has poor scalability, which leads to poor support for the operation of delete or update. Mark et al. [16] proposed an online block-level deduplication technology, sparse indexing, which is based on the similarity detection. This technique divides the data stream into big segments. In order to locate a similar data segment, the technology needs to extract hook fingerprints which can be shared by many blocks. The hook fingerprints of some data blocks may be the same one, and it means that these data blocks are similar to each other. It helps to reduce the sampling rate and lookup complexity. Even though sparse indexing demands less memory than DDFS, its performance of deduplication depends on the locality and sampling rate of hook fingerprint.

2.3. Studies on Transmission Strategies. The authors of paper [19] proposed a cloud backup system, called Cumulus using a simple communication protocol. It only needs to be installed in the client, and the deduplication process is also completed on client, and then Cumulus stores the data block into a remote server. However, the client must maintain a fingerprint index table in order to deduplicate between different files. Though Cumulus has improved the performance of transmission and storage, the system still has a poor scalability. LBFS [20] uses content-based chunking method to divide the files into many chunks. It transmits chunk fingerprints over the network to identify whether a data chunk is duplicated or not. By only transmitting an unduplicated data chunk, LBFS thus saves network bandwidth. The LBFS

runs in user space. Though the LBFS saves bandwidth during transmission, the received files in the storage system are still full-size without any deduplication treatment.

Semantic-aware and application-aware deduplication technology can achieve a tradeoff between deduplication rate and system performance, but this technology is more suitable for personal file backup rather than for virtual machine image management. Enterprise-class deduplication file systems such as DDFS, ZFS, and SDFS demand enterprise-class equipments, while Openstack is usually deployed on limited hardware resources, thus enterprise-class deduplication file systems are not suitable for openstack obviously.

According to the problems and difficulties encountered in image management of Openstack, this paper presented a novel image management system, called IM-dedup system, which uses the static block technology to divide a file into a number of blocks, uses fingerprint pretransmission technology to avoid the transmission of duplicated blocks, deploys the kernel file system with deduplication functionality in the image storage to reduce storage space, uses memory filter to reduce the number of disk indexes, centralizes fingerprint storage area to improve the locality of data accessing, and uses limited memory to achieve a higher IO throughput rate.

The paper is organized as follows. In Section 3.1, the openstack system is introduced. The detailed analysis of image storage and the deduplication process on a sever are given in Section 3.2. In Section 3.3, the IM-dedup system is described and analyzed. In Section 4, the system evaluation methods and results are given. Finally, some conclusions are drawn in Section 5.

3. Design and Implementation of IM-Dedup System

This system takes the advantages of source-deduplication saving bandwidth and target-deduplication saving storage space. It chunks the file data for only once and computes the fingerprint for just once and thus avoids repetitive computation and fully utilizes the existing resources.

3.1. Openstack. Openstack is an open source project developed by NASA and Rackspace, which can be used to build private and public clouds. And it consists of several key components: nova, glance, keystone, swift, network, and so forth. Each component communicates with others by a message queue. Nova is the computation component, which provides application, creation, activation, and destruction services for the virtual machine. Glance is the image management component, which provides storage, query, and registration services for images. Keystone provides registration, management for the account. Swift is the object storage component for Openstack, which provides high-reliability and high-scalability storage service. Network component manages each node's network connection, it can avoid network's bottleneck effectively and improve efficiency. Figure 2 shows the architecture of Openstack.

At present, there are two ways to add an image into openstack.

(1) By glance client and by executing glance add command, the virtual machine image will be added into the storage backend of glance.

(2) By dashboard, users can take a snapshot of a virtual machine and save it as an image.

Method (1) is mostly used by an administrator to manage public virtual machine images.

Method (2) is mostly used by users to manage private virtual machine images.

Every image uploaded to the glance will be assigned the UUID randomly, which is the only identification in the system. No matter which method used, there are two things that openstack must do when storing image:

(1) store a virtual image in the glance,

(2) store the metadata in the glance, which includes registration of the image information in the database, and then mark the image state as saved.

The images of Openstack have 6 states, including queued, saving, active, killed, deleted, and pending_delete. Queued state represents that the UUID of an image is registered into glance, but no image data has been uploaded. Saving state represents that the image data is being uploaded. Killed state represents that there is something wrong during uploading. Active state represents the availability of the image. Delete state represents that image data will be cleaned later; however, the image still can be used at present. Pending_delete state is similar to delete state, but image data can be to reinstated.

Since the needs of users are various, there will be a number of virtual machine images of different versions. As a result, huge storage space is required to store them, and considerable time is needed to transmit them over networks.

3.2. The Process of Server-Side Image Deduplication and Storage. The Openstack should be deployed on the Ubuntu Linux kernel, which uses EXT3 file system [21] to manage files and devices. The hard disk partition of EXT3 file system consists of a series of 4096-byte (4 KB) data blocks; these data blocks are divided into several block groups which are of the same size. Each block group contains a bitmap block (used to record the use of data blocks in the block group), an inode linked list (records each file's inode) and an inode bitmap block (records the use of inode). Each file has a 128-byte inode, which records the creation, modification, access time, size, and save location information of the file. As shown in Figure 3, in the Linux file system, each file will use the data structure called inode, which describes a file or a directory. Each inode contains all of the description information of a file or a directory, including file type, access permissions, file owner, timestamps, file size, data block pointers, and so on. The data pointer can be divided into three types: direct pointers (or the single-level pointers), the second-level pointers, and the third-level pointers. They point to the data from different levels. The three types of pointers are shown in Figure 4.

In order to implement the deduplication and reduce the local storage space of the images, the liveDFS [22] data

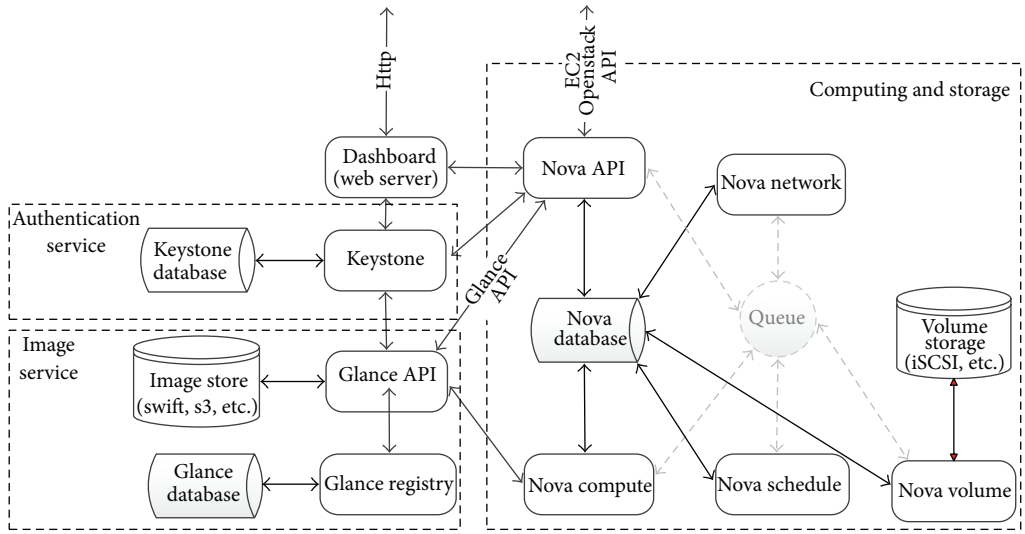


Figure 2: The architecture of Openstack.

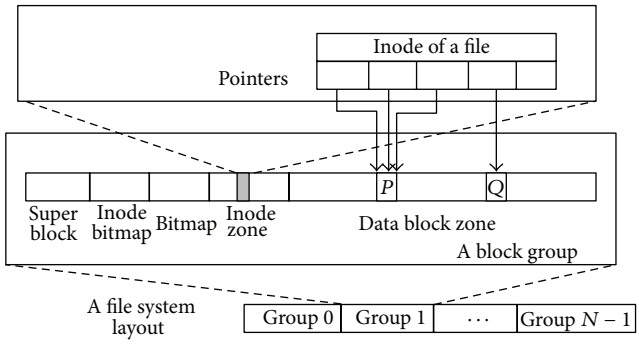


Figure 3: The layout of the file system on the disk [22].

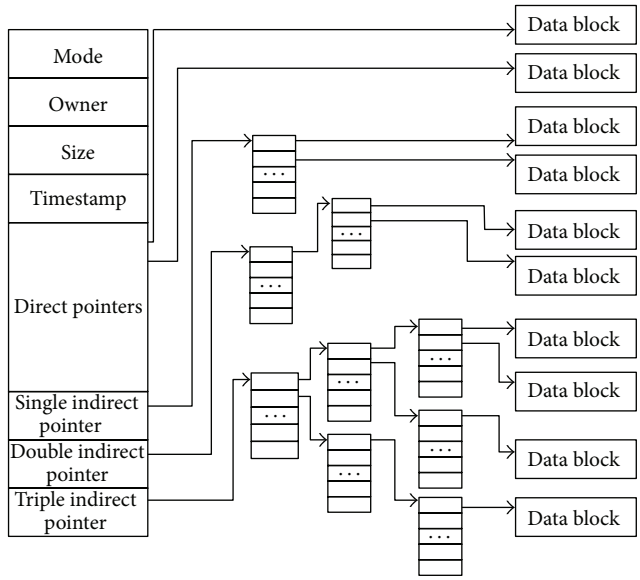


Figure 4: The architecture of an inode in EXT3 file system.

deduplication method is used into the image data storage system on a single computer node. The file system uses a method of fixed block size which can be demonstrated to be capable of optimizing the performance of the CPU and memory. In order to be compatible with liveDFS file system, the IM-dedup system also chooses to use the method of fixed block size. It means that IM-dedup system also divides a file into a number of chunks which size is 4 KB. And the system will extract fingerprint values in a unit of 4 KB. And the system will extract fingerprint values in a unit of 4 KB. For example, file F_1 is cut into C_1, C_2, and C_3, and then the fingerprint values are f_1, f_2, and f_3, respectively; the data blocks C_1, C_2, and C_3 are stored in the file system. F_2 is cut into C_4, C_5, and C_6, and the fingerprint values are f_4, f_5, and f_6, respectively. Finding and comparing through the fingerprint values, if $f_1 = f_5$, then we only stored C_4 and C_6, and let the fingerprint point to C_1, and thus two pointers pointing to one data block, as shown in Figure 5.

3.2.1. Fingerprint Storage. Ext3 file system manages the disk by block groups, and the size of group is 128 MB. And also each file is cut into blocks which size is 4 KB. So, each block group has 32768 blocks in total, and it occupied 655360 bytes space in fingerprints storage, equivalent to the 160 blocks. These 160 blocks exist in the front space of each block group called fingerprints storage area. The fingerprints storage area consists of an array of fingerprints and the reference counter, and it can index to the data block in every block group through each disk data blocks block number (block address) of the same block group, as shown in Figure 5.

One of the fingerprints contains 20 byte, the former 16 byte record fingerprint values of a block and the later 4 byte record the reference number, as shown in Figure 6.

3.2.2. Fingerprint Filter. The fingerprint filter is the index structure in memory. It aims to accelerate the searching speed

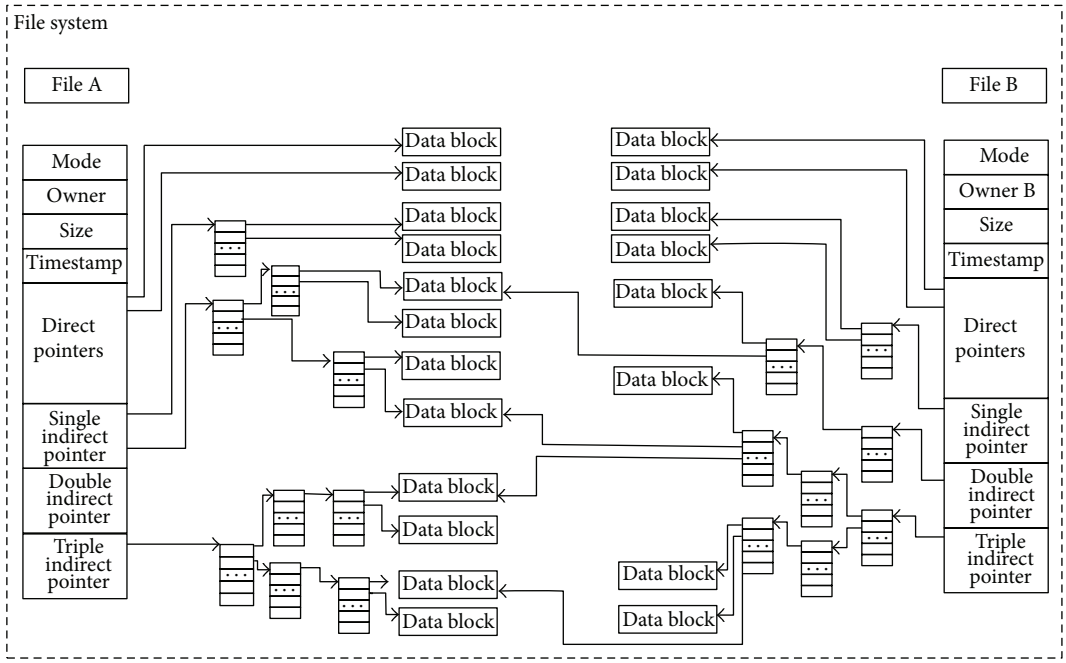


Figure 5: Multiple files sharing data block.

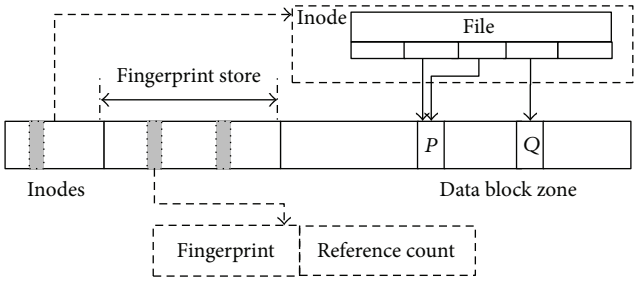


Figure 6: The layout of the a block group [22].

of the fingerprint on disk. Figure 7 describes the fingerprint filter design in detail. The fingerprint filter is the second-level filter. The first-level filter is named index key, which maps the former n bit of fingerprint. The second-level filter is named bucket key, which maps the later k bit of fingerprint.

When initializing the fingerprint filter, Mount program allocates index key table firstly. The index key table is an array of 2^n elements, each element points to the head pointer of a bucket linked list. The tuples which are the same to index key are saved in the bucket which is pointed by this index key. Bucket line is combined with tuples bucket key block number, if one bucket is full, the system will create a new bucket to build bucket linked list. In order to use binary search method to query the bucket key efficiently, we sort the elements in each bucket by bucket key. It needs to be emphasized that the fingerprint filter initialization is a one-time process, and the cost of theinitialization depend on the total amount of data. Before a new block of data is written to the disk, the fingerprint filter is firstly queried. If the new blocks fingerprint is matched with the former $n + k$ bit of the filter, then the filter will return the corresponding data block number. The fingerprint values which share the same prefix "$n + k$" may be more than one, and the filter may return a number of block numbers. In order to eliminate false positives, the system will find the corresponding fingerprint storage area through the return block number to prove if the fingerprint values in data block are exactly matched with the one in fingerprint storage area. After completion of each operation (write, modify, delete), the system will update the fingerprint filter. Ng et al. [22] proved that when $n = 19$ and $k = 24$, the efficiency of the filter can achieve a very good performance through mathematical calculations.

Figure 8 is a new process of writing files.

When a new file comes, system will first chunk it into blocks. And then, the fingerprint calculation program will calculate the fingerprint. Once a fingerprint is produced, it will be transferred into fingerprint filter. If the fingerprint is not in the filter, it indicates that the block is not duplicated. Then the system will write the block into disk and then modify the inode. If the fingerprint is in the filter, the program will start another process into disk to search the fingerprint. If fingerprint is not in the disk, it indicates that the block is a new one. It should be stored into disk, and then some modification should be done to the inode. If fingerprint is in the disk, which means the block is a duplicated one, the block data should be dropped, and what the program needs to do is to modify the pointers in the inode.

3.3. The Design of IM-Dedup System. The infrastructure of IM-dedup system is shown in Figure 9. In Openstack, nova integrates physical resources from compute nodes and provides users computing services. Glance manages the storage, retrieval, and registration of virtual machine image. When a user makes a launch request, nova sends a retrieval request.

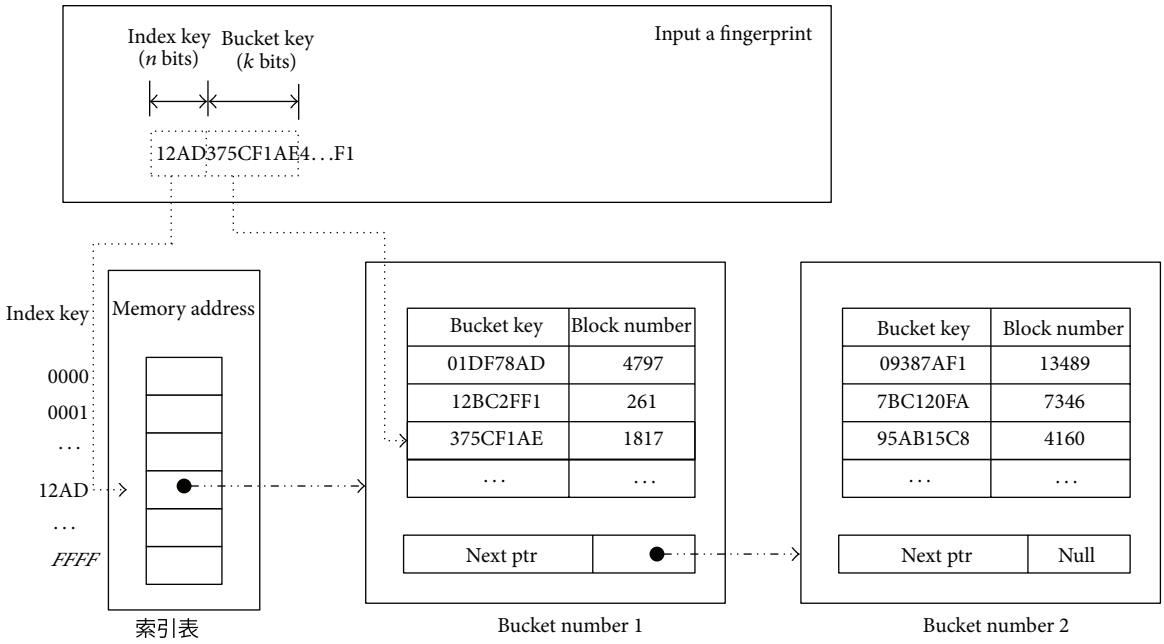


Figure 7: The structure of fingerprint filter [22].

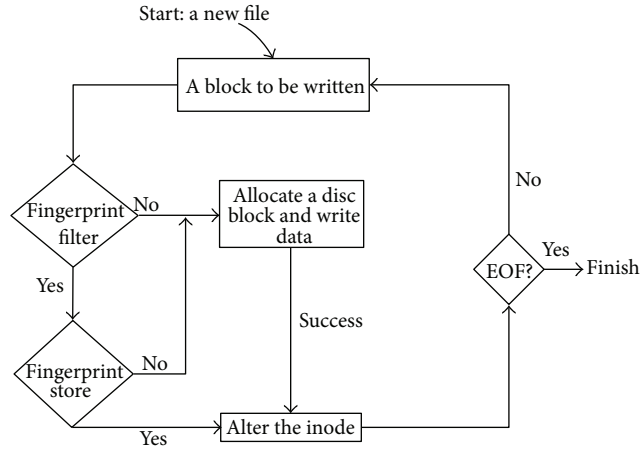


Figure 8: The judging procedure of write data blocks.

And then glance delivers the requested image to a compute node which will then launch this virtual machine image.

When a glance client needs to upload an image, the glance server will register the image information in the registry server and then uploads the image to the storage backend.

3.3.1. The Process of IM-Dedup System. Figure 10 describes the workflow of the process of the image uploading.

Step 1. The glance client sends an image write request to the glance server. The glance server registers the image information to the image database. And the glance server enters the file write mode, makes a new inode, and prepares for the file writing.

Step 2. The glance client starts the slice operation. After getting a chunk c, glance client computes the related fingerprint f and sends it to the glance server.

Step 3. The glance server receives the fingerprint f and then sends it to the fingerprint filter. The fingerprint filter checks f to make sure the $n + k$ bit of f in the filter. If not in, go to Step 6; if in, go to Step 4.

Step 4. Start the process of fingerprint storage retrieval. If f is a new fingerprint, go to Step 5; if not, go to Step 6.

Step 5. If there is f of chunk c the same as chunk c fingerprint f. Modify the present inode pointer to point to c.

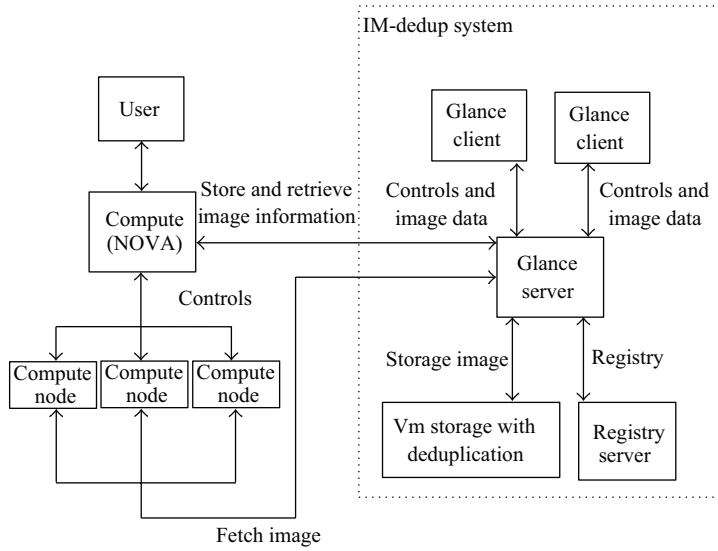


Figure 9: The architecture of IM-dedup system with Openstack.

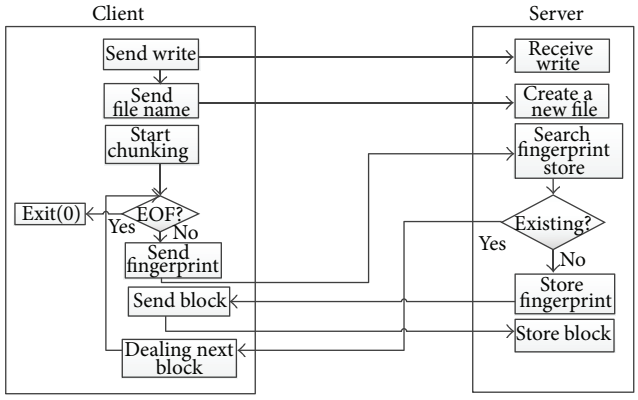


Figure 10: The process of IM-dedup system.

Step 6. Return a request for chunk c to glance client, and then store chunk c onto storage backend, and modify the present pointer to point to chunk c.

Step 7. Go to Step 2, until the end of file.

Step 8. Finish image upload process, and mark it as active.

3.3.2. Chunking. At present, there are two kinds of file-slice methods, including static chunking [23] (SC, as is used in venti [24] and oceanstore [25]) and content defined chunking [11] (CDC, as is used int LBFS [20] and Pastiche [26]). And the deduplication ratio is higher by CDC than by SC [23]. And fingerprint extracting methods widely used are still hash algorithms, like MD5, SHA-1, and Rabin hash.

The present image deduplication technologies only handles single node deduplication, because t the overhead of network data transportation are not considered. To solve this problem, it is necessary to slice the image in the glance client. The present image deduplication technologies only handles single node deduplication, because t the overhead of network data transportation are not considered. Therefore, IM-dedup system adopts SC as the slice method. For seamless combination of native deduplication on image storage server and remote slicing to avoid redundant slicing and fingerprint calculating, IM-dedup system sets chunk size as 4096 bytes.

3.3.3. Methods for Calculating the Data Fingerprint. Each file slicing needs to calculate the data fingerprint. And the system then sends the fingerprint and other metadata to the image storage management server. And on image storage backend, the system will retrieve the fingerprint to make sure that if it is necessary to send this data to image storage backend.

At present, MD5 and SHA-1 are the two most widely used hash algorithms. MD5 groups the input as 512 bits, and the output is 4 32-bits cascades which makes up a 128-bits information abstract. And MD5 is one of the safest encryption algorithms. And SHA-1 generates 160-bits information abstract based on MD5. SHA-1 is a widely used hash algorithm and also the one of most advanced encryption technologies.

4. Experiments

4.1. Experimental Setup. The Openstack Image Service provides discovery, registration, and delivery services for disk and server images. IM-dedup system can do more than Openstack Image Service can. IM-dedup system not only can provide discovery, registration, and delivery services but also provide deduplication and network transmission optimization. The experiments are designed to answer the following 3 questions.

(1) How much storage space saved by IM-dedup system compared to Openstack Image system?

(2) How much transmission time can IM-dedup system save compared to Openstack Image system?

(3) What is the most efficient image formats according to IM-dedup system? In other words, what image formats can reach the highest deduplication rate and get the best transmission optimization results.

In our experimental setup, IM-dedup system is deployed on Lenovo R510 G7 1U server which has an Intel Quad-Core E5606 CPU 2.13 GHz 8 M cache, R-ECC DDR3-1333 4 G memory SAS2.5 inch 300 G disk. The nodes are connected by 1 Gb/S independent Ethernet network.

There are three servers. The first one is used as deduplication image storage server deployed with IM-dedup server. The second one is used as original image storage server deployed with glance server. The last one is used as client, which is in charge of uploading virtual machine image data to servers.

4.2. Data Set. Openstack provides a multiformat image registry; the image service allows uploads of private and public images in a variety of formats, including

(i) Raw (img),

(ii) Machine (kernel/ramdisk outside of image, aka AMI),

(iii) VHD (Hyper-V),

(iv) VDI (VirtualBox),

(v) qcow2 (Qemu/KVM), VMDK (VMWare),

(vi) OVF (VMWare, others).

Some researcher found that the format of virtual machine image impacts a lot on deduplication rate [9]. Images in our data set adopt standardized configuration (2 G memory, 10 G harddisk). Table 3 shows 35 virtual machine images, which contains 7 operating systems and 5 formats. Some of them are in the same lineage (such as Centos.6.2 and Centos.6.3), some of them are in different branches of linux operating system (such as Centos.6.2 and Ubuntu 11.10); some of them are absolutely different (such as Win7 and Centos.6.3). All of them are mainstream operating systems.

4.3. Deduplication Rate. Deduplication rate is an important indicator to the effect of deduplication, which is defined as follows:

$$\text{deduplication rate} = \frac{\text{original bytes of disk image} - \text{stored bytes of disk image}}{\text{original bytes of disk image}}. \quad (1)$$

Deduplication rate depends primarily on deduplication algorithm and deduplication data sets. As some researches [14, 15, 19, 23] show that when the granularity of deduplication increases, the deduplication rate will decrease. But the granularity deduplication decreases, the IO throughput will increase. And it will also reduce CPU efficiency, enhance the memory demand, and decrease overall system performance. At present, the widely adopted deduplication granularity is 8 k. There are some deduplication systems that use 4 M chunking granularity.

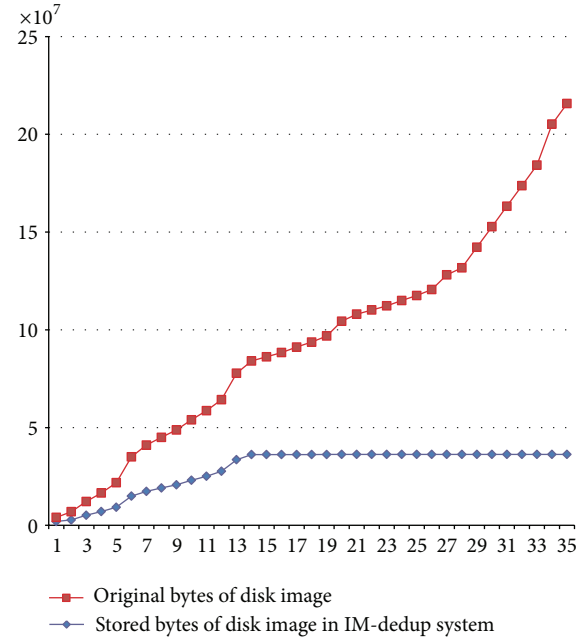


Figure 11: Accumulated storage utilization.

In addition, deduplication rate is also impacted by data sets. For the same data set, the higher deduplication rate the better. In order to test the deduplication performance of IM-dedup system image storage server, this paper uses the original Openstack glance storage backend EXT3 file system as the benchmark.

The accumulated storage utilization of the virtual machine image in the original storage system is shown in the Figure 11. In comparison, when we use the IM-dedup system, with the increase of the number of images, the growth of the storage space slowly and finally leveled off. The reason is that the experimental data sets only using seven image versions (they are Centos.6.2, Centos.6.3, Ubuntu 11.10, Ubuntu 12.04, Ubuntu12.10, WinPro, and Win7), and each image was made into five formats. Although there are 35 images, in fact there are only 7 versions of images, different formats of an operating system may share a plenty of data blocks. Therefore, the deduplication system performs efficiently. The curve of IM-dedup system growth almost decreases to zero. Ideally, if the server stored enough virtual machine image versions, the hit rate of duplicated block will not grow in the server. Because the image server does not need to store duplicated block, the space occupied by virtual machine image will not grow. We can learn from Figure 11 that deduplication rate can be up to 80%.

In order to test the different image formats deduplication rate in the system, this paper selects different system virtual machine images, classified by format, divided into five categories, as shown in Figure 12.

Figure 12 shows that with the same image configuration, raw format occupied the maximum disk space, vmdk, VHD, VDI, and Qcow2 occupied disk space closely. But VHD format image deduplication rate is far lower than the average

Table 3: Images size in the data set (kB).

	Centos. 6.2	Centos. 6.3	Ubuntu 11.10	Ubuntu 12.04	Ubuntu 12.10	Win Pro	Win 7	Totol (kB)
VMDK	2111684	2103556	2768580	2549572	3040452	3587332	7507908	23669084
VHD	2171436	2161196	2853588	2630300	3111700	3615632	7591788	24135640
VDI	2138160	2132016	2799664	2578480	3068976	3612720	7565400	23895416
QCOW2	2107276	2099788	2765772	2546956	3036748	3583940	7507012	23647492
RAW	2100632	2091528	2753424	2536012	3022748	10485764	20971524	43961632
Total (KB)	10629188	10588084	13941028	12841320	15280624	24885388	51143632	

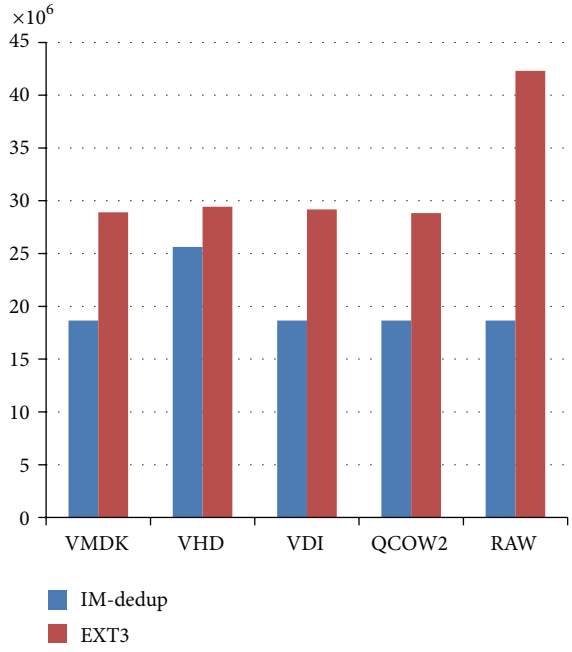


Figure 12: In contrast of the IM-dedup system deduplication effect of virtual machine images of different formats.

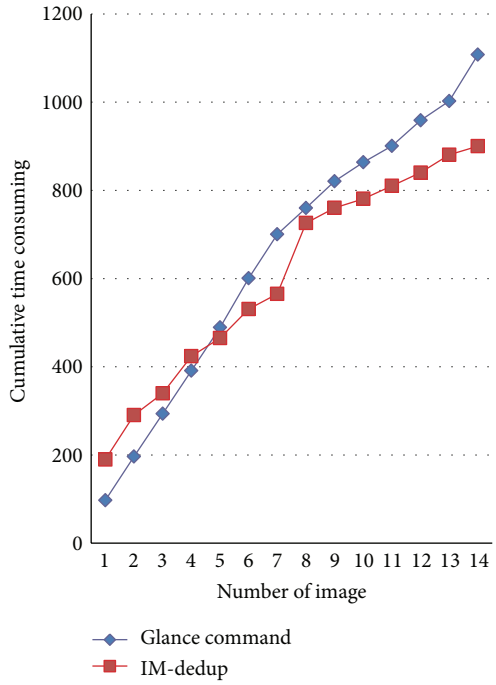


Figure 13: Cumulative time-consuming comparison between glance and IM-dedup system.

deduplication rate 36%. It has only about 13%. Although they have the same conditions on the virtual machine operating system configuration, there are still significant differences between the deduplication rate. This attributes to the internal disk layout of VHD format. But in this paper, there is no in-depth research in this aspect. Experiments show that the openstack supported image formats, VMDK, VDI, QCOW2, and RAW, is ideal for IM-dedup system management, especially the RAW format. Deduplication rate can reach 56%. In practice, the users of the Openstack should recommend the use of these four forms of virtual machine images.

4.4. Network Transfer Optimization. IM-dedup system uses sockets to implement chunk data blocks transmission. With the method of the fingerprint pretransmission, IM-dedup system avoids the transmission of duplicated data blocks. Openstack uses the glance add command to add a new image into image system. Figure 13 shows the time-consuming comparison between glance and IM-dedup system. It can be seen from the chart: when uploading a few images (less than 5 in this experiment), glance performs better than IM-dedup. That is, because glance uploads a whole image without the overhead of chunking, calculation fingerprint and fingerprint index. And there are few images in the image server, so few chunks can be deduplicated. In comparison with glance, IM-dedup system seems to have larger overhead.

However, as the number of images increases (more than 5 in this experiment), the slope of the curve decreases, and cumulative time consuming of IM-dedup system is less than glance. That is, because the number of duplicated data blocks increases as the number of images in storage server increases, and thus more data blocks do not need to be transmitted in IM-dedup.

We also noticed that cumulative time consuming grows very fast when 8th image was being uploaded. Then, we checked the data set and found that the former 7 images are Linux branches (such as Centos, Ubuntu). The 8th image is win7, which shares little duplicated data block with Centos and Ubuntu. The result is also proved in [9].

When the number of upload image grows further, the slope of the IM-dedup curve decreases; however, the slope of

glance does not decrease. In the ideal state, deduplication rate is 100%, and the slope of IM-dedup is infinite toward zero. But in real case, the transmission of requests and fingerprints cannot be neglected. In the best case, we can expect that the time is all consumed by the requests and fingerprints.

Our lab environment is equipped with 1 GB/S Ethernet, which is much better than general conditions. The cost of adding an image by original methods depends on the image size and network condition. When network bandwidth is low, it will consume considerable time. In which case, the advantages of IM-dedup network transport system are more obvious.

After all, the performance of IM-dedup is more prominent when the image number is large and in a low bandwidth environment.

5. Conclusions and Future Work

In this paper, the IM-dedup system based on image management system of deduplication was presented, which could be used in data processing system of distributed wireless sensor networks. The IM-dedup system combined deduplication technology with the original basic functions of openstack, such as glance image upload and registration, to achieve double deduplication of local storage and network transmission. By replacing the original image management system glance in openstack, it achieves better performance. The experimental results showed that IM-dedup system saved at most 80% of the storage space and 30% of image upload time in the image storage server.

This paper mainly studied the way to reduce network transmission time in image upload and achieve deduplication technology in image storage. Future work will be done on the further optimization of image download process by applying deduplication technology.

Acknowledgments

This paper is supported by State Key Development Program of Basic Research of China under Grant no. 2007CB310900, the Hi-Tech Research and Development Program (863) of China under Grant no. 2011AA01A205, Natural Science Fund of China under Grant no. 61202094, no. 61003077, no. 60873023, and no. 60973029, the Science and Technology Major project of Zhejiang Province (Grant no. 2011C11038), Zhejiang Provincial Natural Science Foundation under grant no. Y1101104, Y1101092, and Y1090940. Zhejiang Provincial Education Department Scientific Research Project (no. Y2-01016492).

References

[1] L. Youyou, A. Li, and S. Jiwu, "A remote data backup system with deduplication," *Journal of Computer Research and Development*, vol. 2012, S1, pp. 206–210, 2012.

[2] L. Qiang, Z. Ligu, and Z. Fu, "The desktop backup technology based on cloud storage," in *Proceedings of the China National Computer Conference (CNCC '10)*, CCF, HangZhou, China, 2010.

[3] M. Jian-ting and Y. Pin, "Deduplication-based file backup system for multiuser," *Computer Engineering and Design*, vol. 32, no. 11, pp. 3586–3589, 2011.

[4] W. Zhen, *The research on deduplication software technology in backup system of bank [Ph.D. thesis]*, FuDan University, Shanghai, China, 2009.

[5] Y. Lawrence and K. Christos, "Evaluation of efficient archival storage techniques," in *Proceedings of the Conference on Mass Storage Systems and Technologies (NASA/IEEE MSST '04)*, pp. 227–232, Greenbelt, Md, USA, 2004.

[6] Eucalyptus project, 2012, http://www.eucalyptus.com/.

[7] EC2, 2012, http://aws.amazon.com/ec2/.

[8] Openstack project, 2012, http://www.openstack.org/.

[9] J. Keren and L. Ethan Miller, "The effectiveness of deduplication on virtual machine disk images," in *Proceeding of the Israeli Experimental Systems Conference (SYSTOR '09)*, Article 7, New York,USA, 2009.

[10] A. Liguori and E. Van Hensbergen, "Experiences with content addressable storage and virtualdisks," in *Proceedings of the Workshop on I/O Virtualization (WIOV '08)*, San Diego, Calif, USA, 2008.

[11] Z. Benjamin, L. Kai, and P. Hugo, "Avoiding the disk bottleneck in the data domain deduplication file system," in *Proceedings of the Conference on File and Storage Technologies (FAST '08)*, San Jose, Calif, USA, 2008.

[12] OpenDedup, 2012, http://code.google.com/p/opendedup/downloads/detail?name=SDFScture.pdf.

[13] OpenSolaris, 2012, http://hub.opensolaris.org/bin/view/Community+Group+zfs/dedup.

[14] Y. Fu, H. Jiang, N. Xiao, L. Tian, and F. Liu, "AA-Dedupe: an application-aware source deduplication approach for cloud backup services in the personal computing environment," in *Proceedings of the IEEE International Conference on Cluster Computing (CLUSTER '11)*, pp. 112–120, Austin, Tex, USA, September 2011.

[15] Y. Tan, H. Jiang, D. Feng, L. Tian, Z. Yan, and G. Zhou, "SAM: a semantic-awaremulti-tiered source de-duplication framework for cloud backup," in *Proceedings of the 39th International Conference on Parallel Processing (ICPP '10)*, pp. 614–623, San Diego, Calif, USA, September 2010.

[16] L. Mark, K. Eshghi, D. Bhagwat, V. Deolalikar, G. Trezis, P. Camble et al., "Sparse indexing: large scale, inline deduplication using sampling and locality," in *Proceedings of the 7th USENIX Conference on File and Storage Technologies (FAST '09)*, pp. 24–27, San Francisco, Calif, USA, 2009.

[17] S. Rhea, R. Cox, and A. Pesterev, "Fast, inexpensive content addressed storage in Foundation," in *Proceedings of the USENIX Annual Technical Conference on Annual Technical (ATC '08)*, pp. 143–156, Berkeley, Calif, USA, 2008.

[18] D. Bhagwat, K. Eshghi, D. D. E. Long, and M. Lillibridge, "Extreme binning: scalable, parallel deduplication for chunk-based file backup," in *Proceedings of the IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS '09)*, pp. 237–245, London, UK, September 2009.

[19] V. Michael, S. Savage, and G. M. Voelker, "Cumulus: file system Backup to the Cloud," in *Proceedings of the Conference on File and Storage Technologies (FATS '09)*, San Francisco, Calif, USA, 2009.

[20] M. Athicha, B. Chen, and D. Mazières, "A low-bandwidth network file system," in *Proceedings of the 18th ACM Symposium on*

Operating Systems Principles (SOSP '01), New York, NY, USA, 2001.

[21] G. XiMei, *An in-depth research on filesystem and disk management mechanism [Unpublished M.Phil thesis]*, Nanjing University of Aeronautics and Astronautics, Jiangsu, China, 2002.

[22] C.-H. Ng, M. Ma, T. -Y. Wong, P. P. C. Lee, and J. C. S. Lui, "Live deduplication storage of virtual machine images in an open-source cloud," in *Proceedings of the ACM/IFIP/USENIX 12th International Middleware Conference*, Lisboa, Portugal, 2011.

[23] D. Meister and A. Brinkmann, "Multi-level comparison of data deduplication in a backup scenario," in *Proceedings of the Israeli Experimental Systems Conference (SYSTOR '09)*, Article 8, May 2009.

[24] Q. Sean and D. Sean, "Venti: a new approach to archival storage," in *Proceedings of the Conference on File and Storage Technologies (FAST '02)*, San Jose, Calif, USA, 2002.

[25] J. Kubiatowicz, D. Bindel, Y. Chen et al., "OceanStore: an architecture for global-scale persistent storage," in *Proceedings of the 9th Internatinal Conference Architectural Support for Programming Languages and Operating Systems (ASPLOS-IX)*, pp. 190–201, Cambridge, Mass, USA, November 2000.

[26] L. P. Cox, C. D. Murray, and B. D. Noble, "Pastiche, making backup cheap and easy," in *Proceedings of the 5th Symposium on Operating System Design and Implementation*, Boston, Mass, USA, 2002.

[27] X. Shushen, *The research and application with hash algorithm [Unpublished M.Phil thesis]*, Hubei University of Technology, Hubei, China, 2006.

Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2013, Article ID 867903, 15 pages
http://dx.doi.org/10.1155/2013/867903

Research Article

Energy-Efficient Bridge Detection Algorithms for Wireless Sensor Networks

Orhan Dagdeviren and Vahid Khalilpour Akram

International Computer Institute Bornova, Ege University, 35100 Izmir, Turkey

Correspondence should be addressed to Orhan Dagdeviren; orhan.dagdeviren@ege.edu.tr

Received 27 January 2013; Accepted 8 April 2013

Academic Editor: Hongju Cheng

A bridge is a critical edge whose fault disables the data delivery of a WSN component. Because of this, it is important to detect bridges and take preventions before they are corrupted. Since WSNs are battery powered, protocols running on WSN should be energy efficient. In this paper, we propose two distributed energy-efficient bridge detection algorithms for WSNs. The first algorithm is the improved version of Pritchard's algorithm where two phases are merged into a single phase and radio broadcast communication is used instead of unicast in order to remove a downcast operation and remove extra message headers. The second algorithm runs proposed rules on 2-hop neighborhoods of each node and tries to detect all bridges in a breadth-first search (BFS) execution session using $O(N)$ messages with $O(\Delta(\log_2(N)))$ bits where N is the node count and Δ is the maximum node degree. Since BFS is a natural routing algorithm for WSNs, the second algorithm achieves both routing and bridge detections. If the second proposed algorithm is not able to to classify all edges within the BFS phase, improved version of Turau's algorithm is executed as the second phase. We show the operation of the algorithms, analyze them, and provide extensive simulation results on TOSSIM environment. We compare our proposed algorithms with the other bridge detection algorithms and show that our proposed algorithms provide less resource consumption. The energy saving of our algorithms is up to 4.3 times, while it takes less time in most of the situations.

1. Introduction

Rapid developments in the last decade in wireless and hardware technologies have created low-cost, low-power multifunctional miniature wireless devices. These devices have enabled the use of wireless sensor networks (WSNs) [1]. WSNs do not have any fixed infrastructure where hundreds even thousands of sensor nodes cooperate to implement a distributed application. WSNs can be used in various applications including habitat monitoring [1], military [2], and smart home applications [3]. Energy consumption of a WSN should be reduced to maximize the application lifetime since sensor nodes are battery powered. The radio component of a sensor node is the dominant energy consumer part.

WSNs are increasingly being used in challenged environments such as underground mines, tunnels, oceans, and the outer space. Wireless communication in challenged environments have channel (edge) failures, mainly as a consequence of direct impact of physical world. In addition to energy constraints and wireless communication problems, tiny sensor motes are prone to failures. In both type of these failures, sensor network can continue its operation without a serious bad effect. On the other side, some edges can have critical tasks in routing operation. These edges are called bridges (cut edges) which its removal breaks connectivity of the network. Bridge detection is an important research area for different types of networks [4–9]. After bridges are detected, various solutions can be applied [10] in order to neutralize bridges.

A WSN can be modeled with an undirected graph $G = (V, E)$, where V is the set of nodes and E is the set of edges. BFS and depth-first search (DFS) are fundamental graph traversal algorithms. DFS starts from sink node, and searches deeper in the graph if possible [11]. Like DFS, BFS starts from sink node and search proceeds in a breadth-first manner [11]. The edges chosen for BFS are called tree edges, and other edges are called cross edges. In BFS, tree level of sink node is 0, the levels of neighbors of sink node are 1, and levels of other

nodes are their shortest distance to the sink node. From this property, BFS can be used to construct shortest path trees rooted at the sink node. BFS is used widely in sensor network for various purposes such routing and localization [12–16]. Besides, BFS can be modified to detect bridges. In this study, we proposed bridge detection algorithms that use BFS as the basis algorithm.

Distributed bridge detection algorithms proposed so far have some important disadvantages. Although distributed DFS based algorithms [8, 17–20] are simple and efficient for bridge detection, DFS based applications for WSN are rare in practice. Since then, DFS should be implemented as a separate service where this would be an extra load for battery-powered sensor nodes. Although BFS provides an efficient routing infrastructure for sensor networks, BFS based bridge detection algorithms lack some important design criterions. The message size of the Milic's BFS based algorithm [9] can be as large as $O(E \log_2(N))$. Pritchard's BFS based algorithm has two extra phases [21]. Because of these reasons, these algorithms may consume significant energy. Regarding these deficiencies, we propose two distributed localization-free and energy-efficient bridge detection algorithms for sensor networks. The contributions of this paper are listed below.

(i) We propose an improved version of Pritchard's algorithm (I-PRITCHARD). In I-PRITCHARD algorithm, radio broadcast communication is used instead of unicast communication, and a downcast operation in Pritchard's algorithm is removed. I-PRITCHARD can be completed within 2 phases; on the other hand, Pritchard's algorithm needs 3 phases. Because of these reasons, I-PRITCHARD consumes less energy than the Pritchard's algorithm.

(ii) We propose the energy-efficient bridge detection algorithm (ENBRIDGE) by using 2-hop neighborhood knowledge and radio broadcast communication. The algorithm uses $O(N)$ messages with $O(\Delta \log_2(N))$ bits, where N is the node count and Δ is the maximum node degree. Besides, the algorithm runs just one phase, that is, integrated with the BFS at the best case. This is a significant improvement over Milic's algorithm. In the worst case, the algorithm runs an improved DFS algorithm where message complexity and message size are asymptotically same with the first phase. ENBRIDGE outperforms its counterparts in the simulations.

The rest of this paper is organized as follows. In Section 2, the network model and bridge detection problem are described, and the related work is surveyed in Section 3. The proposed algorithms are described in Section 4. The detailed analysis of the algorithms are given in Section 5, and the results of performance tests are presented in Section 6. Conclusions are given in Section 7.

2. Background

In this section, we introduce the network model and the bridge detection problem.

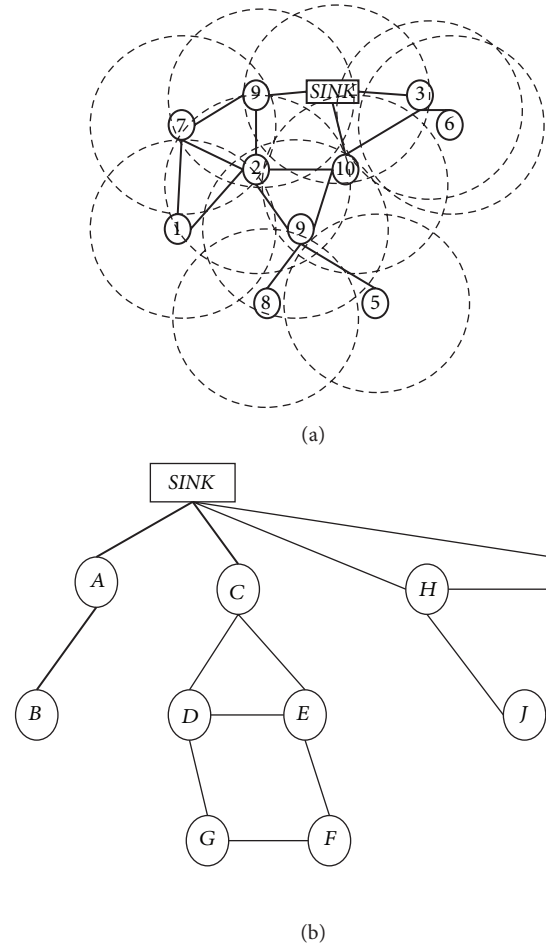


Figure 1: (a) Undirected graph model. (b) The bridge problem.

2.1. Network Model. The following assumptions are made about the network as in [8, 22].

(i) Each node has distinct *node_id*.

(ii) The nodes are stationary.

(iii) Links between nodes are symmetric. Thus, if there is a link from u to v, there exists a reverse link from v to u.

(iv) Nodes do not know their positions. They are not equipped with a position tracker like a GPS receiver.

(v) All nodes are equal in terms of processing capabilities, radio, battery, and memory.

(vi) Each node can send radio broadcast messages to its immediate neighbors in its transmission range.

(vii) Nodes are time synchronized. This can be achieved by implementing a time synchronization protocol as proposed in [23].

Based on these assumptions, the network may be modeled as an undirected graph $G(V, E)$, where V is the set of vertices and E is the set of the edges. An example of undirected graph model is depicted in Figure 1(a), where the

transmission ranges of the nodes are shown with dashed circles.

2.2. The Bridge Detection Problem. Bridges can connect any two nodes on the network. A bridge can connect a leaf node to its parent or connect a whole component to lower layers. An example of sensor network is depicted in Figure 1(b). There are 10 nodes, where node ids are written inside of each node, and the communication edges are shown with the solid lines in this Figure. The edges (*A*, *B*), (*A*, *SINK*), and (*C*, *SINK*) are bridges which are depicted with bold lines. If the edge (*A*, *B*) fails then leaf node *B* cannot transmit its packet to the lower layers. If the edge (*A*, *SINK*) fails then both nodes *A* and node *B* cannot relay their data packets where 20% of the network can not transmit its data to the sink node. Node *C* is the parent of a component consisting of the node *D*, the node *E*, the node *G*, and the node *F*. If the edge (*C*, *SINK*) fails, 50% of the total nodes cannot send their data to the sink node. In this paper, our focus is energy-efficient bridge detection for sensor networks, so our objectives are listed below.

(1) Since message transmission is the dominant factor of energy consumption, the bridge detection algorithm should be efficient in terms of message complexity and message size.

(2) Routing is a fundamental operation for sensor networks. It is crucial for data delivery and data aggregation [24]. If the bridge detection can be integrated with the routing operation, it can introduce a less total cost to the network.

(3) Sink node may initiate the bridge detection algorithm locally. Hence, these operations should be distributed.

(4) Bridge detection algorithm should be independent from the underlying protocols as much as possible to interface to various MAC and physical layer standards such as in [25–30].

(5) The algorithm should not be dependent on localization information.

3. Related Work

DFS algorithm can be centrally executed by the sink node in order to detect bridges [4, 9]. Since collecting the whole network information is expensive and not always possible, various distributed implementations are proposed for DFS algorithm [17, 20, 31–36]. Most of these algorithms can be modified to detect bridge in sensor networks by using the rules proposed by Tarjan [4]. Cidon [17], Hohberg [18], Chaudhuri [19], Tsin [20], and Turau [8] proposed distributed DFS algorithms for bridge and cut vertex (a vertex whose removal breaks the connectivity of a graph) detection algorithms. Turau's algorithm [8] is an extended and sensor network adopted version of Cidon's [17] and Tsin's [20] algorithms. At the worst case, Turau's algorithm transmits $4E$ messages with $O(\log_2(N))$ size, where E is the number of edges and N is the total node count. Since unicast messages are used in Turau's algorithm, if the medium access control (MAC) layer does not provide an edge based sleep schedule, the upper bound for the received and overhead messages is $O(\Delta E)$. Our algorithms have $\Theta(N)$ sent message complexity, and $O(\Delta N)$ received and overheard message complexity. Also we show in this paper with extensive simulations that our algorithms are practically favorable.

Like DFS, BFS can be centrally executed to detect bridges [9]. Although this algorithm is simple to implement, execution of central BFS is an expensive operation in terms of energy consumption caused by message transfers, and it is not suitable for large scale self-organizing distributed sensor networks. Because of these reasons, distributed BFS algorithms are proposed [12–16]. For synchronous networks, a well-known greedy algorithm is applied to construct BFS [14]. This algorithm consumes $O(N)$ messages and $O(D)$ time, where D is the diameter of the network. Although this algorithm is very effective for constructing routing infrastructure, it is not adequate to find bridges without any extension. Liang proposed a BFS algorithm for biconnectivity testing algorithm which runs on permutation graphs and which cannot be generalized [5]. Thurimella proposed a BFS biconnectivity testing algorithm in which each processor is assumed to know the whole topology [6]. Because of this property, algorithm is not suitable for WSNs.

Milic proposed a bridge detection for wireless ad hoc networks [9]. The algorithm uses broadcast communication of wireless nodes, and it is integrated with the BFS operation. The forward phase of the algorithm is nearly the same with the standard BFS operation. In the backward phase of the algorithm, the nodes store a list of cross edges that they found or received, append cross edges to the messages, and send to their parents. Although the algorithm completes its operation within a BFS execution interval, the message size can be very large since it is dependent on the number of cross edges. The message size can be as large as $O(E \log_2(N))$. On the other side, our proposed algorithms' message size are $O(\Delta \log_2(N))$. Besides, in this paper, we simulate Milic's algorithm against various network topologies and show that our proposed algorithms are favorable.

Pritchard proposed a three-phased algorithm for the distributed bridge detection [21]. In the first step, the algorithm finds a spanning tree by implementing distributed greedy BFS tree algorithm. In the second step, the algorithm computes subtree sizes and preorder labels. In the last step, bridges are detected. The time complexity of the algorithm is $O(D)$, the message complexity is $O(E)$, and the message size is $O(\log_2(N))$ bits. In this paper, we first propose an improved version of this algorithm. Secondly, we propose an energy-efficient bridge detection algorithm which can be integrated with the BFS operation and can finish within the BFS execution. The algorithms covered so far exactly find bridges without localization. In this study, we omit localization-based bridge detection algorithms such as [16].

4. Proposed Algorithms

4.1. Improved PRITCHARD Algorithm. PRITCHARD algorithm effectively detects bridges of an undirected graph

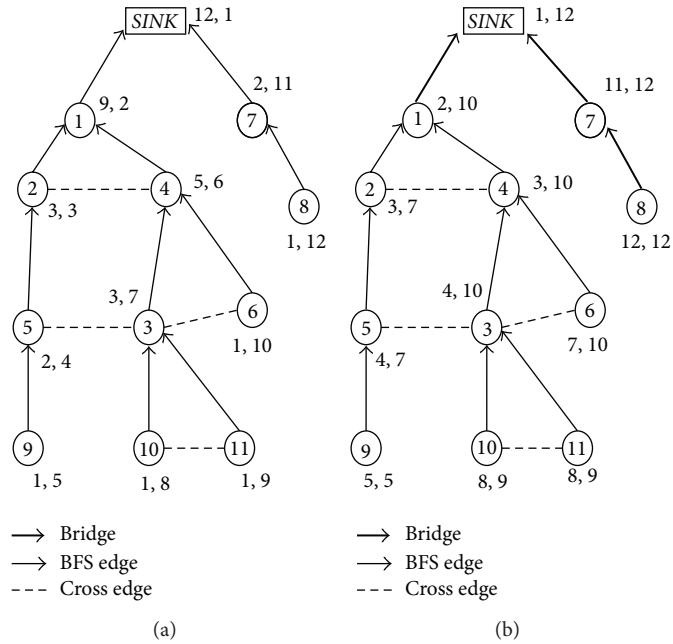


Figure 2: (a) I-PRITCHARD Phase 1 (b) I-PRITCHARD Phase 2.

by applying a 3 phase method. Although the algorithm is well designed, it can be further improved for battery-powered sensor nodes. To achieve this, we propose the I-PRITCHARD algorithm which includes the following list of modifications in order to reduce transmitted message counts and transmitted bit counts.

(1) Phase 1 and Phase 2 can be merged into a single phase. When the nodes are executing backward operation during BFS execution in Phase 1, each node may calculate its subtree size (*#desc*), and the convergecast operation in Phase 1 can be accomplished. To achieve this, each message in backward phase should include *#desc* field which is $O(\log_2(N))$ bits size. After backward operation is finished, Phase 2 is not executed, so that $O(N)$ messages which are flooded by the initiator node are saved by applying this improvement.

(2) Preorder messages in Phase 2 and announcement messages in Phase 3 are sent as broadcast messages instead of unicast. In this case, although the message size increases to $O(\Delta\log_2(N))$ bits, header fields for separate messages are not transmitted. So total transmitted bit counts are reduced.

An example operation is given in Figure 2. Since BFS execution is integrated with the preorder labeling in I-PRITCHARD, the algorithm is executed in 2 phases. The first phase is depicted in Figure 2(a). In this figure, ID of the nodes is written inside of each node, *#desc* value and preorder label are written near to each node. The second phase is depicted in Figure 2(b). In this figure, low and high values are written near to each node. The edges (1, *SINK*), (7, *SINK*), (5, 9), and (7, 8) are the bridges.

4.2. ENBRIDGE Algorithm. The algorithms covered so far have below listed deficiencies which motivate us to design ENBRIDGE algorithm.

(i) Even though the design of an energy-efficient DFS based bridge detection algorithm is possible, DFS applications are rare in real-world applications. Because of that, the DFS based bridge detection module may not be integrated to the other applications.

(ii) Although MILIC may be easily integrated to the BFS, transmitted bit count is proportional to the network diameter and cross edge count.

(iii) PRITCHARD and I-PRITCHARD are BFS based algorithms, transmitted bit counts are proportional to the average neighbor degree. Although these algorithms are energy-efficient, extra phases are executed after the BFS algorithm.

We propose ENBRIDGE algorithm for detecting bridges in WSN in an energy-efficient manner. The algorithm has two phases. In the first phase, an extended BFS algorithm is executed. The forward phase of the BFS algorithm is the same. At the backward phase of the BFS algorithm, each node broadcasts its edge states to its neighbors where a state of an edge can be one of *CHILD*, *CROSS*, and *BRIDGE*. Hence, each node knows the edge states of its neighbors. By using these 2-hop information, each node $n \in V$ runs the ENBRIDGE_Classify given in Algorithm 1 to check whether its edge connecting to the parent node p is bridge. Each rule given in Algorithm 1 is executed sequentially by the nodes since they are ordered by considering their computational complexity. The computational complexity of the first three rules is $O(\Delta)$, and the computational complexity of the last rule is $O(\Delta^3)$.

(1) // Algorithm inputs 2-hop neighbors of node n.
if the node n has only 1 incident edge **then** the edge (n, p) is a bridge (*Rule 1*).
(2) **else if** the node n has at least one cross edge **then** the edge (n, p) is not a bridge (*Rule 2*).
(3) **else if** the edge (n, p) is the only edge connecting node n to lower levels and all other edges are bridges **then** the edge (n, p) is a bridge (*Rule 3*).
(4) **else if** one of node n's children has a cross edge connecting to node x where node x's parent is not n and id of one of node n's children **then** the edge (n, p) is not a bridge (*Rule 4*).
(5) **else if** all neighbors of node n's children are also children of node n **then** the edge (n, p) is a bridge (*Rule 5*).
(6) **end if**.
(7) **if** one of these rules are applied **then** return true.
(8) **else** return false.
(9) **end if**.

Algorithm 1: ENBRIDGE BFS edge classification algorithm (ENBRIDGE_Classify).

After executing these rules at the backwards stage of the first phase, each node notifies whether it is able to classify its parent link. To achieve this notification, a 1-bit *classified* field is transmitted during convergecast operation in backwards stage. If the node n or one of descendants cannot classify its parent link, *classified* field gets 0, otherwise it gets 1. Inductively, sink node finds whether one of the BFS edges is left unclassified. If the node n sends classified as false, then its parent node p does not execute ENBRIDGE_Classify as given in Algorithm 2 in order to save CPU power. If all BFS edges are classified, then sink node stops the execution of the algorithm. Otherwise, the sink node starts the second phase of the ENBRIDGE algorithm.

Although the second phase of the ENBRIDGE algorithm is not always executed, it should be energy-efficient as the first phase. For the second phase, we propose an improved version of the TURAU's DFS based algorithm (I-TURAU). In this improved version, *SEARCH* messages are sent as broadcast messages instead of unicast messages so *VISITED* messages no longer need to be transmitted. With this improvement, $O(N)$ messages with $O(\log_2(N))$ bits are transmitted during the second phase of the ENBRIDGE algorithm. The second phase can classify all edges in all situations where the first phase cannot. From this fact, one can claim that the execution of the first phase is redundant. Although this claim can be true for sensor networks which do not use BFS like routing infrastructures, this claim is false for sensor networks where BFS is the dominant routing protocol. The detailed ENBRIDGE algorithm is given in Algorithm 2.

Example operations of ENBRIDGE are depicted in Figure 3. In the first example shown in Figure 3(a), all edges can be classified by the first phase of the algorithm. Edges (10, 12) and (6, 11) are classified as bridges by executing Rule 1. Edges (1, 3), (2, 4), (4, 6), (5, 7), (5, 8), and (5, 9) are classified as nonbridges by executing Rule 2. Edge (6, 10) is classified as bridge by executing Rule 3. Edges (1, *SINK*) and (2, *SINK*) are classified as nonbridges by executing Rule 4. Edge (3, 5) is classified as bridge by executing Rule 5. Since all edges can be classified, the second phase is not executed, and the ENBRIDGE algorithm is finished. In the second example given in Figure 3(b), although edges (1, 4), (2, 5), (4, 6), and (5, 10) can be classified as nonbridges, (1, 2) and (1, *SINK*) can not be classified. Thus, the second phase is executed. In the second phase, (1, 2) is classified as nonbridge, and (1, *SINK*) is classified as bridge edge.

5. Analysis

In this section, we will analyze proof of correctness, message, time, space, and computational complexities of the I-PRITCHARD and ENBRIDGE algorithms.

5.1. Proof of Correctness

Theorem 1. *Nodes executing I-PRITCHARD detect bridges and terminate the execution correctly.*

Proof. Correctness of merging Phase 1 and Phase 2 into a single phase can be proved by induction. Since the BFS execution is synchronous, each leaf node can calculate its *#desc* as 1 during backward phase of BFS as the base case of the induction. Each nonleaf node can calculate its *#desc* by aggregating its children's *#desc*, where this operation is continued until the sink node's execution inductively. The proof of correctness of using broadcast instead of unicast is trivial since the same information is received by all nodes. □

Theorem 2. *Each node detects its parent link state correctly after executing the ENBRIDGE algorithm.*

Proof. We first prove the correctness of the ENBRIDGE_Classify algorithm. To prove the correctness of Rule 1, we assume that the node n has only 1 incident edge. In this case, the node n is a leaf; thus, (n, p) is a bridge. To prove the correctness of Rule 2, we assume that the node n has a cross edge e, and then e can be one of followings.

(i) The edge e equals (n, x) where the node x's level (level_x) is smaller then or equal to the level_n. In both of this cases, the edge (n, p) is not a bridge since the node n has an incident edge other than (n, p) which connects the node n to lower layers.

(ii) The edge e equals (n, x) where the level_x is greater than level_n. Since the node x is not a child of the node

```
(1) message formats: FORWARD(source, level, parent), BACKWARD(source, destination, edges, classified)
(2) initially: id is the unique node identifier; parent ← ∞; children ← ∅; crosses ← ∅;
    neighbors is the set of collected sources of HELLO messages; forward_sent ← false;
    forward_received ← false; two_hop_edges ← ∞; classified ← true
(3) sink node multicasts FORWARD(0, 0, ∞) to neighbors
(4) upon a node receives FORWARD(source, p_level, p_parent) message
(5)   forward_received ← true
(6)   if parent = ∞ then
(7)     parent ← id; level ← p_level + 1; assign the edge (source, id) as a BFS edge
(8)   else if p_parent = id then
(9)     children ← children ∪ source; assign the edge (source, id) as a BFS edge
(10)  else assign the edge (source, id) as a cross edge; crosses ← crosses ∪ source
(11)  end if
(12) end upon
(13) upon a new period starts
(14)  if forward_sent = false ∧ forward_received = true then
(15)    forward_sent ← true; multicast FORWARD(id, level, parent) to neighbors
(16)    if neighbors = children ∪ crosses ∪ {parent} then
(17)      classified ← ENBRIDGE_Classify(assigned edges)
(18)      send BACKWARD(id, parent, assigned edges, classified) to parent
(19)    end if
(20)  else if ∀ BACKWARD messages from i ∈ children received then
(21)    two_hop_edges ← two_hop_edges ∪ assigned edges
(22)    if classified = true then
(23)      classified ← ENBRIDGE_Classify(two_hop_edges)
(24)    end if
(25)    if id = sink then
(26)      if classified = true then finish execution
(27)      else start I-TURAU execution
(28)      end if
(29)    else
(30)      send BACKWARD(id, parent, assigned edges, classified) to parent
(31)    end if
(32) end upon
(33) upon a node receives BACKWARD(source, destination, edges, p_classified)
(34)  two_hop_edges ← two_hop_edges ∪ edges; classified ← classified ∧ p_classified
(35) end upon
```

Algorithm 2: ENBRIDGE main algorithm.

n, then the $\text{level}_{\text{parent}_x}$ can be at most level_n. Thus, same as in previous case, (n, p) is not a bridge.

To prove the correctness of Rule 3, we assume that the node n does not have any cross edge; the edge (n, p) is the only edge connecting node n to lower layers, and all other edges are bridges. In this case, the node n does not have any alternative path connecting it to lower layers which excludes the edge (n, p), so the edge (n, p) is a bridge. To prove the correctness of Rule 4, we assume that one of the node n's children (node c) has a cross edge connecting to node x, where $parent_x$ is not equal to the n and one of node n's children. In this case, an alternative path can be found as (n, c), (c, x), and (x, m) as shown in Figure 4(a). To prove the correctness of Rule 5, we assume that Rule 2 and Rule 4 are not true and all neighbors of the node n's children are also children of the node n. In this case, since all edges other than the edge (n, p) cannot connect the node n to the lower layers, the edge (n, p) is a bridge. An instance of this case is depicted in Figure 4(b).

If all of these rules are not applicable, ENBRIDGE uses broadcast-based TURAU to find bridges. Thus, ENBRIDGE detects bridges, and execution of the ENBRIDGE terminates in all nodes. □

5.2. Message Complexity

Theorem 3. *The count of sent messages in I-PRITCHARD is 3N-1 at the best case and 4N-3 at the worst case.*

Proof. At the best case, the nodes are arranged as a star topology, where $2N$-1 messages are sent at the first phase, 1 message is sent by the center node at the beginning of the second phase, and N-1 messages are sent by the other nodes at the end of the second phase. Thus, $3N$-1 total messages are sent at the best case. At the worst case, $2N$-1 messages are sent at the first phase, N-1 announcement messages are sent at the beginning of the second phase, and N-1 at the end of the second phase; thus, $4N$-3 messages are sent. □

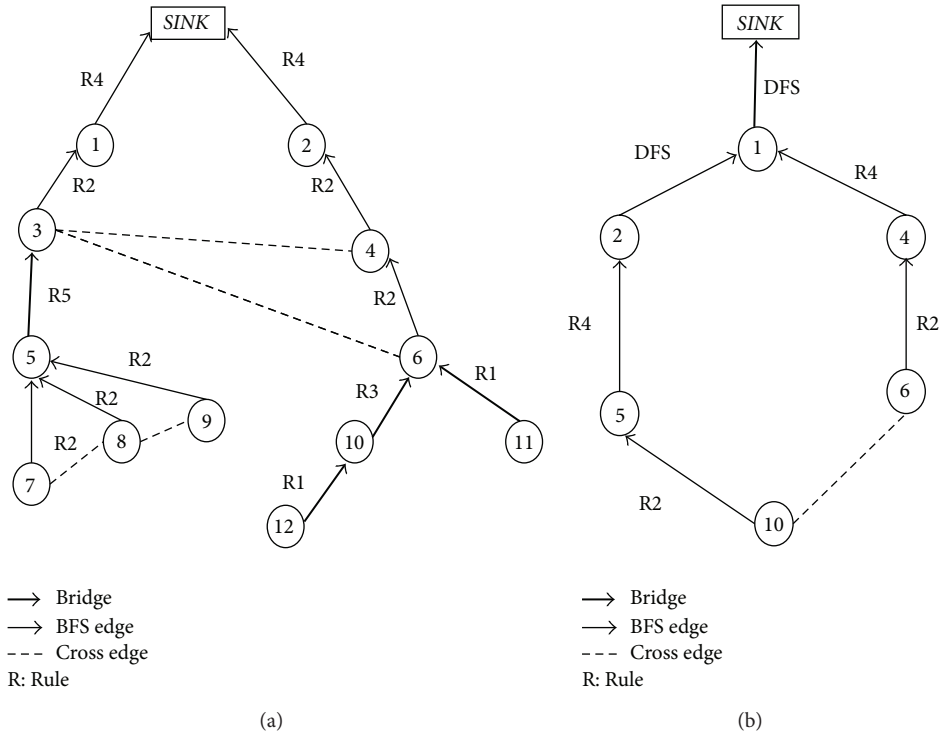


Figure 3: (a) ENBRIDGE Phase 1 classifies all edges and (b) ENBRIDGE Phase 2 is necessary.

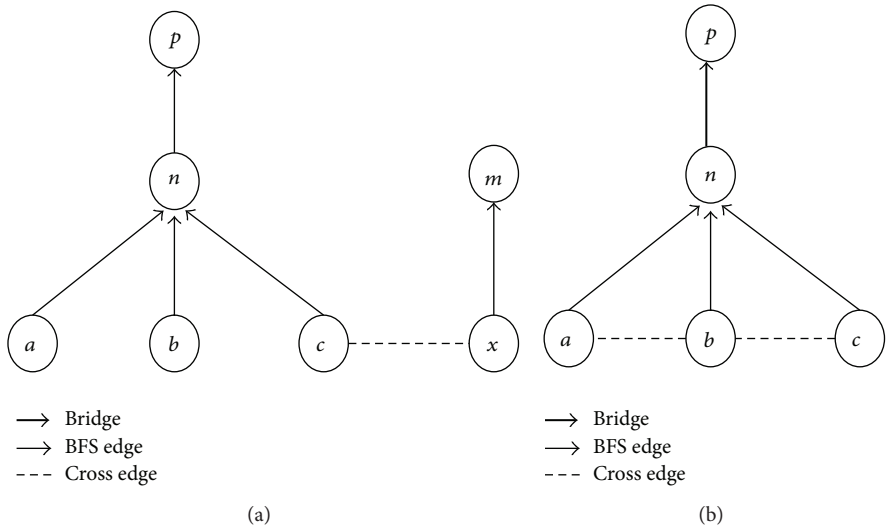


Figure 4: (a) Example for ENBRIDGE Rule 4. (b) Example for ENBRIDGE Rule 5.

Theorem 4. *The count of sent messages in ENBRIDGE is 2N-1 at the best case and 4N-3 at the worst case.*

Proof. At the best case, only BFS is executed on the star topology, so $2N$-1 messages are sent. At the worst case, an extra DFS is executed, where each node uses broadcast instead of unicast, so $2N$-2 messages are sent. Hence, $4N$-3 total messages are sent at the worst case. □

Theorem 5. *The count of received and overheard messages of I-PRITCHARD is O(ΔN).*

Proof. At the worst case, each node receives and overhears Δ messages at the first and second phases. Thus, total count for N nodes is O(ΔN). □

Theorem 6. *The count of received and overheard messages of ENBRIDGE is O(ΔN).*

Table 1: Analytical comparison of algorithms.

Algorithm/complexity	Sent messages	Received messages	Message size	Time	Space (per node)	Computational (per node)
CENTRAL	$\Theta(N^2)$	$O(\Delta N^2)$	$O(\Delta \log_2(N))$	$O(D)$	$O(E)$	$O(E)$
MILIC	$\Theta(N)$	$O(\Delta N)$	$O(E \log_2(N))$	$O(D)$	$O(E)$	$O(E)$
TURAU	$\Theta(E)$	$\Omega(\Delta N)$, $O(\Delta^2 N)$	$O(\log_2(N))$	$O(N)$	$O(\Delta)$	$O(\Delta^2)$
PRITCHARD	$\Theta(E)$	$\Omega(\Delta N)$, $O(\Delta^2 N)$	$O(\log_2(N))$	$O(D)$	$O(\Delta)$	$O(\Delta)$
I-PRITCHARD	$\Theta(N)$, lower:$3N-1$, upper:$4N-3$	$O(\Delta N)$	$O(\Delta \log_2(N))$	$\Omega(D)$, $O(N)$	$O(\Delta)$	$O(\Delta)$
ENBRIDGE	$\Theta(N)$, lower:$2N-1$, upper:$4N-3$	$O(\Delta N)$	$O(\Delta \log_2(N))$	$\Omega(D)$, $O(N)$	$O(\Delta^2)$	$\Omega(\Delta)$, $O(\Delta^3)$

Proof. Each node receives and overhears Δ *FORWARD* and *BACKWARD* messages during BFS operation at the worst case. This bound is the same for the DFS operation, so that total count of received and overheard messages of ENBRIDGE for N nodes is $O(\Delta N)$. □

Theorem 7. *The message size of I-PRITCHARD is $O((\Delta \log_2(N))$ bits.*

Proof. In I-PRITCHARD, each parent node announces pre-order label of its children by broadcasting a single message. Thus, the message size of the I-PRITCHARD is $O(\Delta \log_2(N))$ bits. □

Theorem 8. *The message size of ENBRIDGE is $O(\Delta \log_2(N))$ bits.*

Proof. At the backwards stage of ENBRIDGE, each node broadcasts its incident edge states to its neighbors. Because of this, the message size of the ENBRIDGE is $O(\Delta \log_2(N))$ bits. □

5.3. Time, Space, and Computational Complexities

Theorem 9. *The time complexity of I-PRITCHARD is $O(D)$.*

Proof. Since proposed improvements do no effect on time complexity, time complexity of I-PRITCHARD algorithm depends on the network diameter. Because of this, the time complexity of I-PRITCHARD is $O(D)$. □

Theorem 10. *ENBRIDGE takes $\Omega(D)$ time at the best case and $O(N)$ time at the worst case.*

Proof. At the best case, only Phase 1 is executed, so that the time complexity of the ENBRIDGE algorithm is equal to the time complexity of the BFS operation, so that the lower bound of the time complexity is $\Omega(D)$. At the worst case, Phase 2 is executed with phase 1. In this case, the time complexity is equal to the worst case time of the DFS operation, so that the upper bound of the time complexity is $O(N)$. □

Theorem 11. *The space and computational complexities of the I-PRITCHARD algorithm is $O(\Delta)$.*

Proof. Each node should store its neighbor's state where this table can be at most $O(\Delta)$. The algorithm executes on this table, so computational complexity is $O(\Delta)$. □

Theorem 12. *The space complexity of ENBRIDGE is $O(\Delta^2)$.*

Proof. Each node should store its 2-hop neighbor's state, so that this table can be at most $O(\Delta^2)$. □

Theorem 13. *The computational complexity of the ENBRIDGE algorithm is $O(\Delta^3)$. The lower bound for the computational complexity is $\Omega(\Delta)$.*

Proof. At the best case, one of Rule 1, Rule 2, and Rule 3 is executed which results in the $\Omega(\Delta)$ computational complexity. In order find the computational complexity of Rule 4, we assume that the node n has c cross edges which are represented as (a, b) and which are not incident to it but incident to one of its children (lets call it a). In order to find whether a is not equal to n or one of n's children, $c(\Delta^2 - c)$ computations should be made. If we maximize this equation, then $c = \Delta/\sqrt{2}$, and the computational complexity of Rule 4 becomes $O(\Delta^{5/2})$. Execution of Rule 5 can be $O(\Delta^3)$ at the worst case since a node may have Δ^2 2-hop neighbor nodes, and these 2-hop neighbor nodes are searched in the list of Δ 1-hop neighbor nodes. □

A summary and analytical comparison of central algorithm (CENTRAL), Milic's Algorithm (MILIC), Turau's Algorithm (TURAU), Pritchard's Algorithm (PRITCHARD), I-PRITCHARD, and ENBRIDGE algorithms are given in Table 1. MILIC, I-PRITCHARD, and ENBRIDGE are asymptotically better algorithms in terms of sent and received messages. The message sizes of I-PRITCHARD and ENBRIDGE are $O(\Delta \log_2(N))$; on the other side, message size of MILIC is $O(E \log_2(N))$. Although TURAU and PRITCHARD algorithm's message sizes are $O(\log_2(N))$ bits, the sent and received message counts are higher than other algorithms. Since ENBRIDGE algorithm's lower bound of sent message count is $2N$-1, it is analytically the best algorithm among other algorithms in terms of energy consumption caused by sent and received messages. ENBRIDGE is favorable in terms of energy consumption, but its time and computational complexities are worse than I-PRITCHARD as

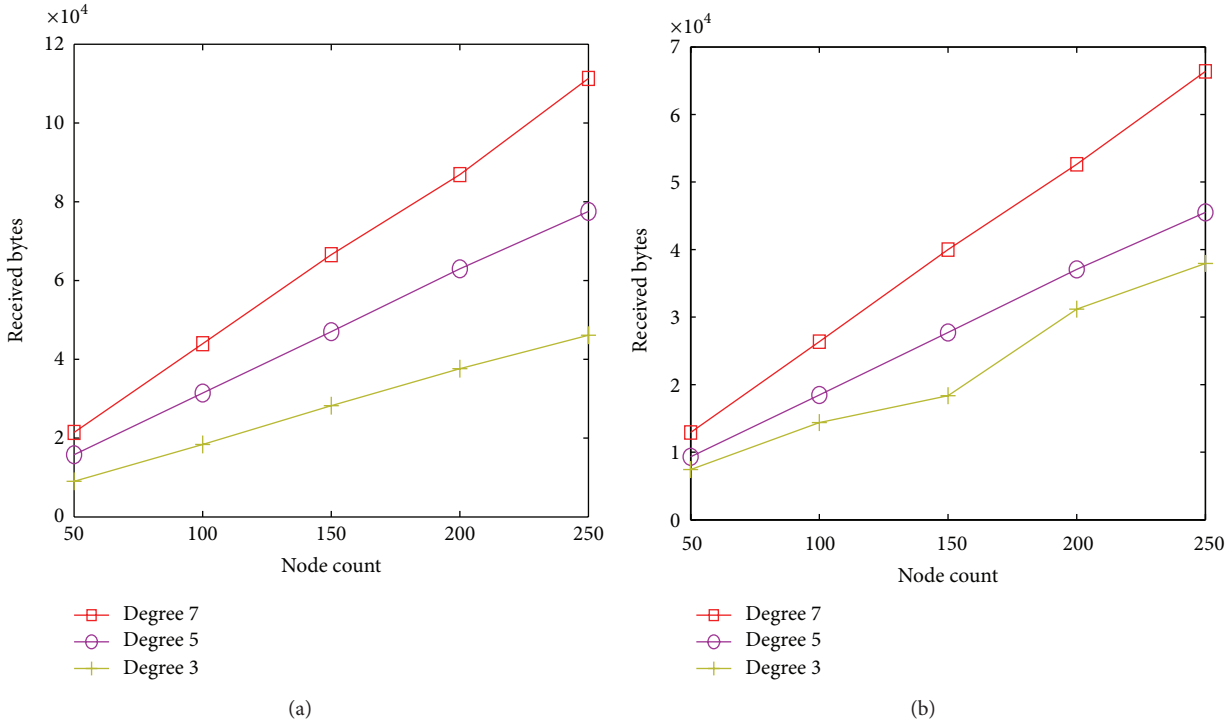


Figure 5: (a) Received bytes of I-PRITCHARD against node degree and node count. (b) received bytes of ENBRIDGE against node degree and node count.

shown in Table 1. Although the worst case time complexity of ENBRIDGE is worse than I-PRITCHARD, its best case time complexity is equal to I-PRITCHARD's time complexity. Besides, ENBRIDGE may terminate just after BFS execution, where I-PRITCHARD has an extra phase.

6. Performance Evaluations

We implemented I-PRITCHARD and ENBRIDGE algorithms in TOSSIM simulator [37] to evaluate their performance. TOSSIM simulator is developed by the researchers from University of California Berkeley, Intel Research Berkeley, and Harvard University. TOSSIM inherits the TinyOS's structure and provides a simulation environment that is close to the real world. The nesC compiler which is also currently used for the TinyOS applications is modified in order to use the same compiler for both simulation and implementation. A discrete event queue is used in the execution model. By using this queue, all simulator events are timestamped and processed in order. The hardware parts of the real-world implementations are emulated in TOSSIM. Two radio models are simulated. In the first model, the developers use error-free transmission to test the correctness of their protocols. The second model provides the developers to test the multihop transmissions.

We implemented CENTRAL, MILIC, TURAU, and PRITCHARD algorithms in order to compare them with the proposed algorithms. We generated randomly connected networks varying from 50 to 250 nodes that are uniformly distributed over the sensing area. For the lower layers, we implemented a TDMA based MAC protocol, and we used the IEEE 802.15.4 physical layer. The transmission power is 3 dBm. The average degrees of the nodes in generated networks are varying between 3, 5, and 7. We measured total received bytes, total sent bytes, energy consumption, and wallclock times. Each measurement is the average of 10 repeated simulations. In each simulation, nodes are randomly placed in the area. Table 2 summarizes the simulation parameters.

Table 2: Simulation parameters.

Node distribution	Random
Sink position	Randomly placed in the area
Number of sensors	50–250
MAC	TDMA
Transmission power	3 dBm
Transmission range	50 m
Node degrees	3, 5, and 7

6.1. Received Byte Counts. Since radio transmitter is the dominant energy consumer of the component of the sensor node, received byte counts are important evaluation criterions. Until otherwise stated, the default node degree is 5, and the node count is 150. Total received byte counts of I-PRITCHARD and ENBRIDGE against node count and node degree are shown in Figures 5(a) and 5(b), respectively. When the node degree and node count are increased, total received byte count of both algorithms increase linearly.

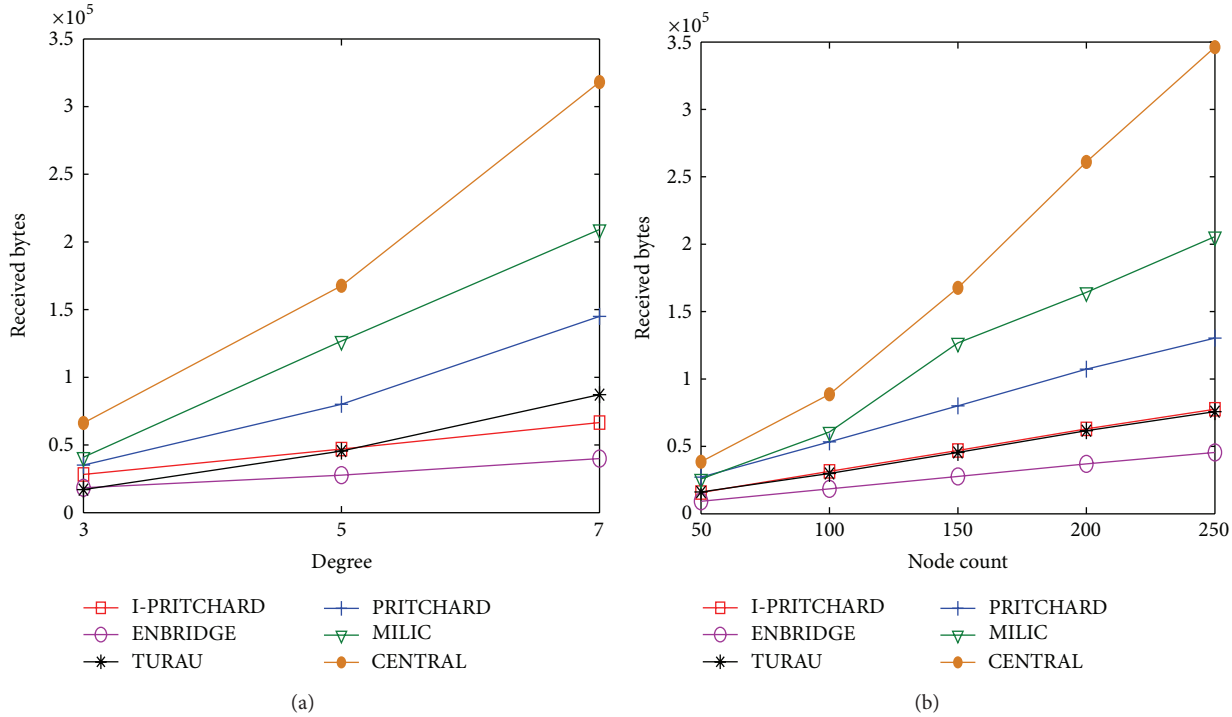


Figure 6: (a) Received bytes of algorithms against node degree. (b) Received bytes of algorithms against node count.

These results conform with theoretical analysis where the complexity of received and overheard messages is $O(\Delta N)$. These results show that I-PRITCHARD and ENBRIDGE are scalable against varying node degrees and node counts.

Performance comparisons of I-PRITCHARD, ENBRIDGE, and their counterparts are given in Figures 6(a) and 6(b). The received byte count of ENBRIDGE is the smallest, and I-PRITCHARD is the second smallest among the other algorithms. The received byte count performance order of algorithms is ENBRIDGE, I-PRITCHARD, TURAU, PRITCHARD, MILIC, and CENTRAL. These results show that BFS based approaches other than I-PRITCHARD and I-MILIC are worse than Turau's DFS based approach. I-PRITCHARD is approximately 1.7 times better than PRITCHARD on the average. ENBRIDGE is approximately 2 times better than TURAU, 3 times better than PRITCHARD, 4.5 times better than MILIC, and 6 times better than CENTRAL algorithm on the average. The reasons of this significant performance of proposed algorithms are using broadcast messages with at most $O(\Delta \log_2(N))$ bits.

6.2. Sent Byte Counts. Secondly, we measured the sent byte counts to evaluate the energy consumption of the algorithms. Total sent byte counts of I-PRITCHARD and ENBRIDGE against node count and node degree are shown in Figures 7(a) and 7(b), respectively. Total received byte count values fluctuate between similar values when the node degree and node count are increased. These results show that I-PRITCHARD and ENBRIDGE are stable and scalable against varying node degrees and node counts.

Total sent byte counts of I-PRITCHARD, ENBRIDGE, and their counterparts are given in Figures 8(a) and 8(b). The sent byte counts of ENBRIDGE are the smallest, and those of I-PRITCHARD are the second smallest among the other algorithms similar to the received byte count performance. The sent byte count performance order of algorithms is ENBRIDGE, I-PRITCHARD, TURAU, PRITCHARD, MILIC, and CENTRAL. The sent byte count of I-PRITCHARD approximately is 1.7 times smaller than PRITCHARD on the average. ENBRIDGE is approximately 1.8 times better than PRITCHARD, 3.6 times better than MILIC, 4.5 times better than MILIC, and 8 times better than CENTRAL algorithm on the average.

6.3. Energy Consumption. Energy efficiency is one of the most important objective for WSN. We measured the energy consumption of the distributed bridge detection algorithms. We assumed that the energy consumption occur mostly by message transfers. Energy consumptions of I-PRITCHARD and ENBRIDGE against node count and node degree are shown in Figures 9(a) and 9(b), respectively. Energy consumptions increase linearly when the node degree and node count are increased. These results show that the energy consumption of I-PRITCHARD and ENBRIDGE are stable and scalable against varying node degrees and node counts.

Performance comparisons of I-PRITCHARD, ENBRIDGE, and their counterparts are given in Figures 10(a) and 10(b). The ENBRIDGE algorithm consumes 3.4 mJ per node, and it has the best performance. The I-PRITCHARD algorithm consumes 5.3 mJ per node, and it has the second performance among the other algorithms. I-PRITCHARD

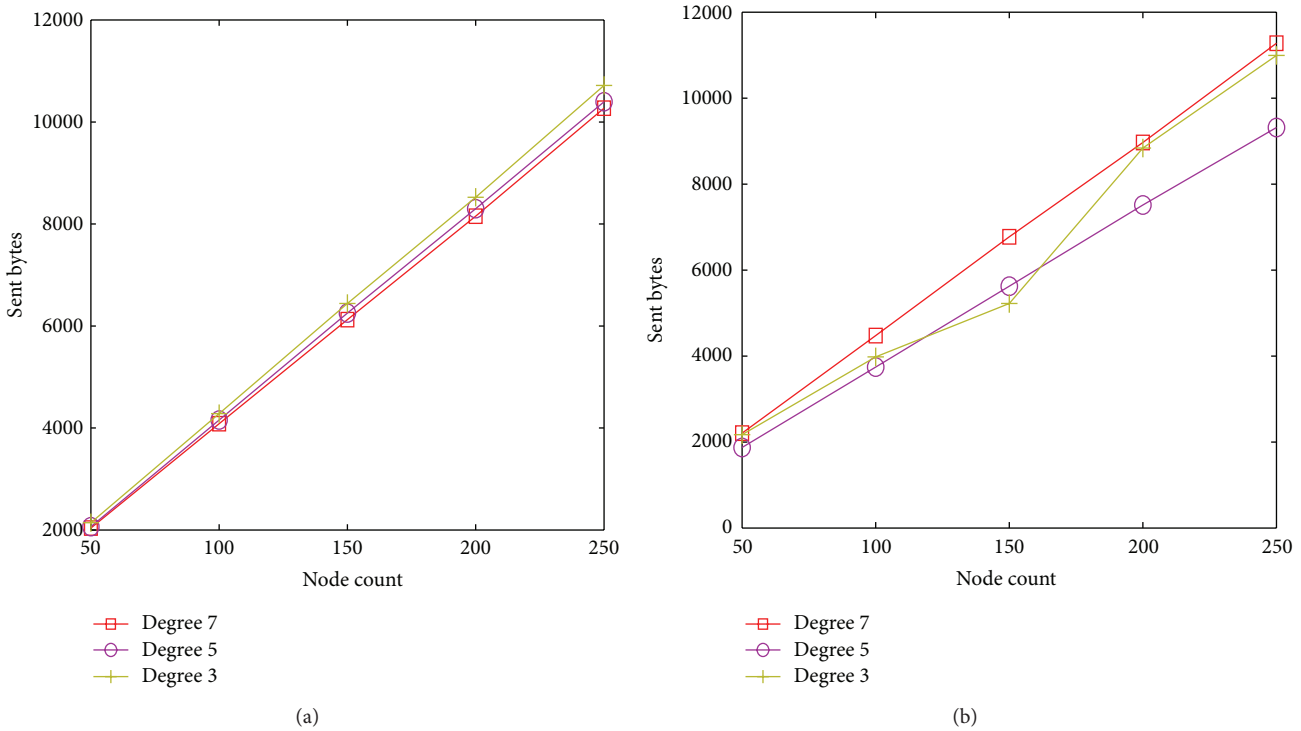


FIGURE 7: (a) Sent bytes of I-PRITCHARD against node degree and node count. (b) Sent bytes of ENBRIDGE against node degree and node count.

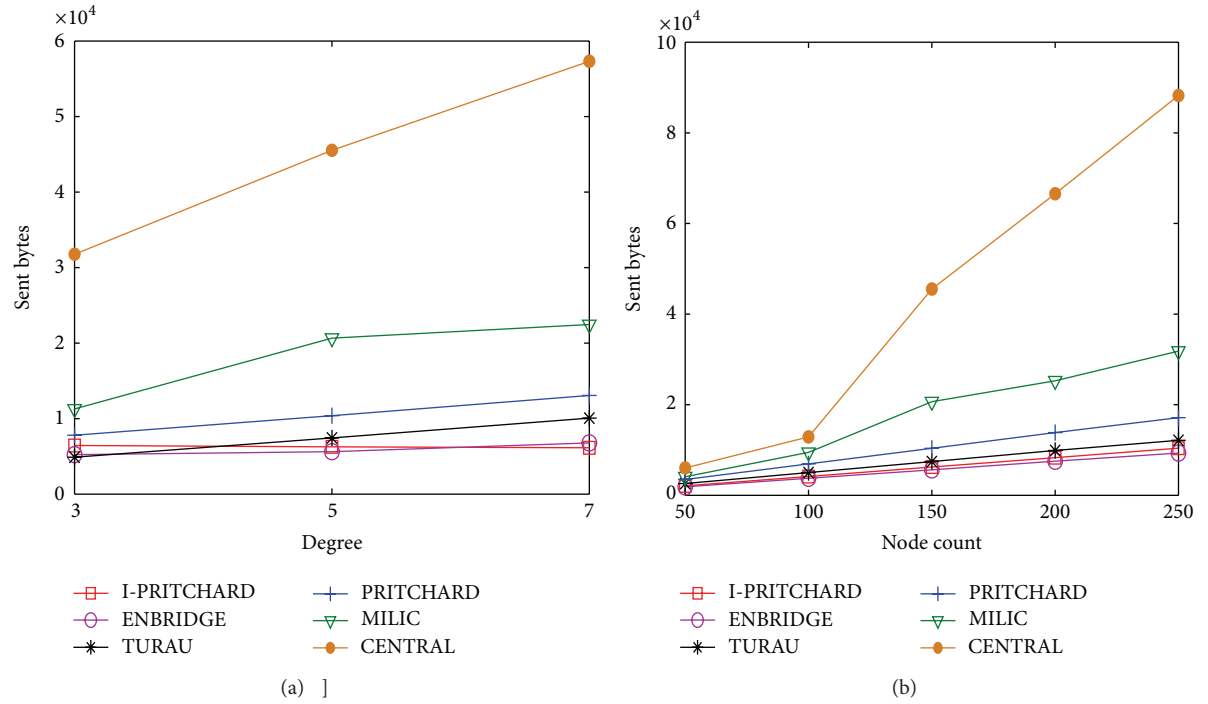


FIGURE 8: (a) Sent bytes of algorithms against node degree. (b) Sent bytes of algorithms against node count.

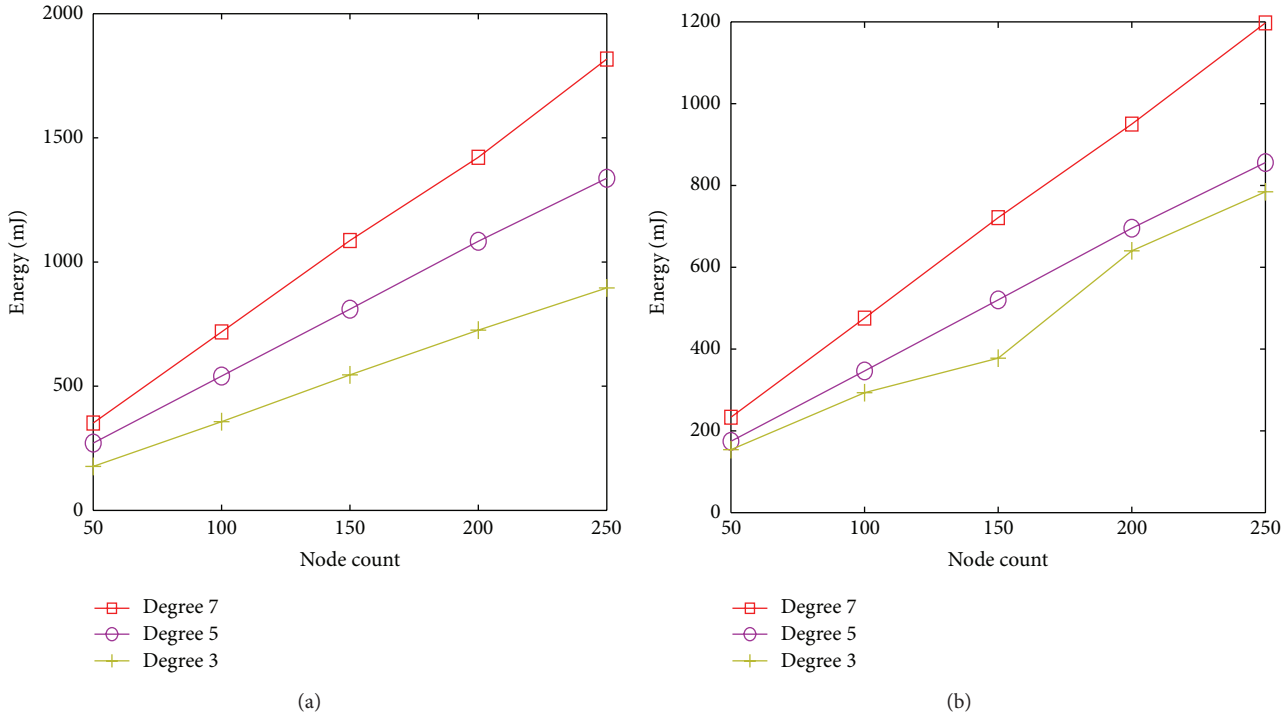


FIGURE 9: (a) Energy consumption of I-PRITCHARD against node degree and node count. (b) Energy consumption of ENBRIDGE against node degree and node count.

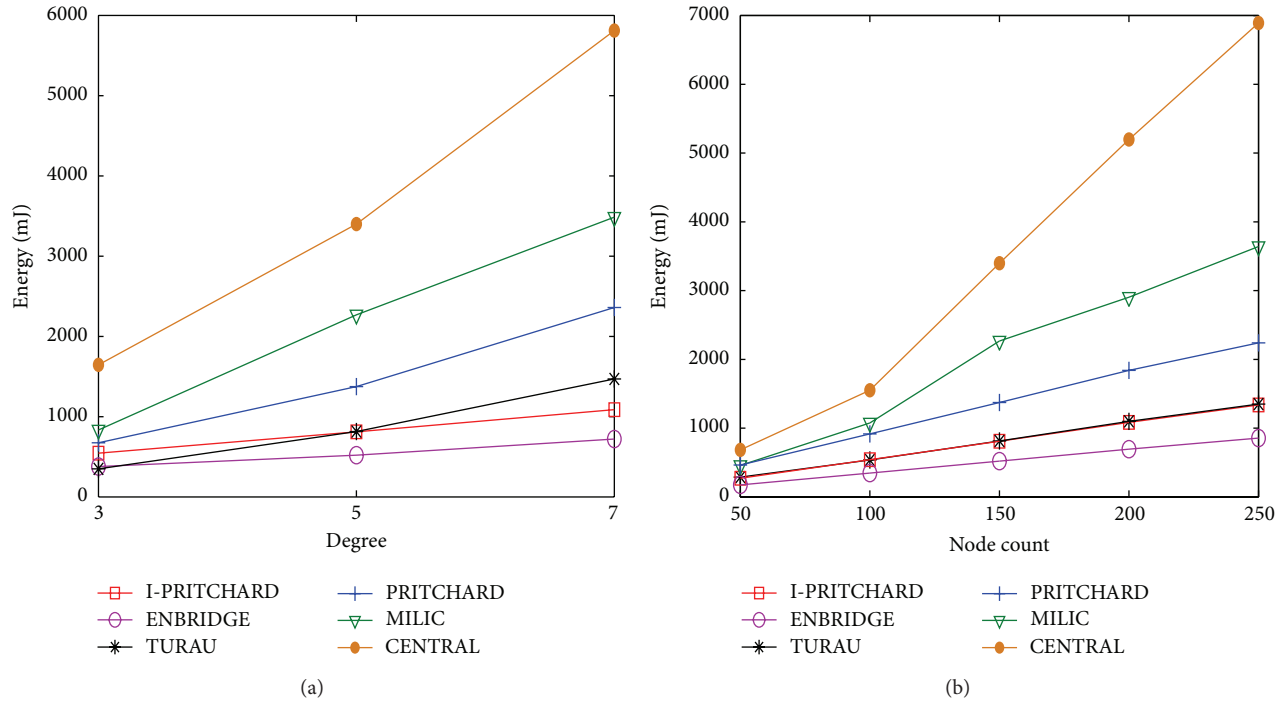


FIGURE 10: (a) Energy consumption of algorithms against node degree. (b) Energy consumption of algorithms against node count.

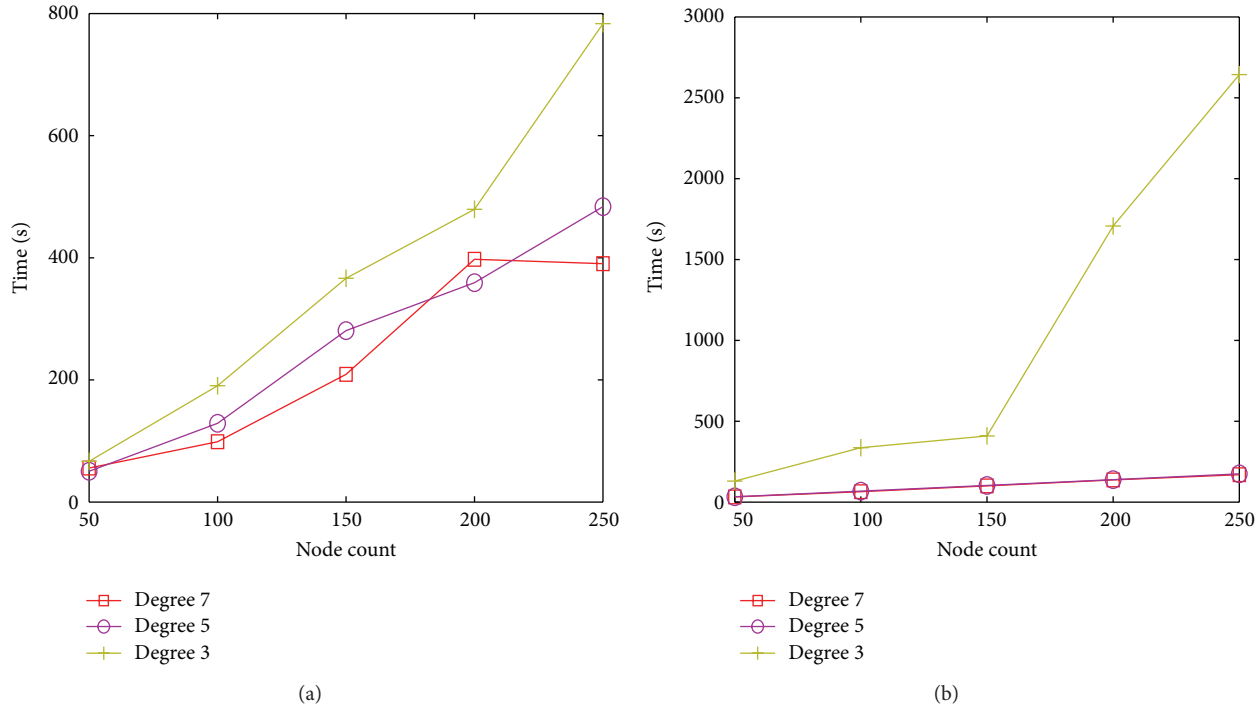


Figure 11: (a) Wallclock times of I-PRITCHARD against node degree and node count. (b) Wallclock times of ENBRIDGE against node degree and node count.

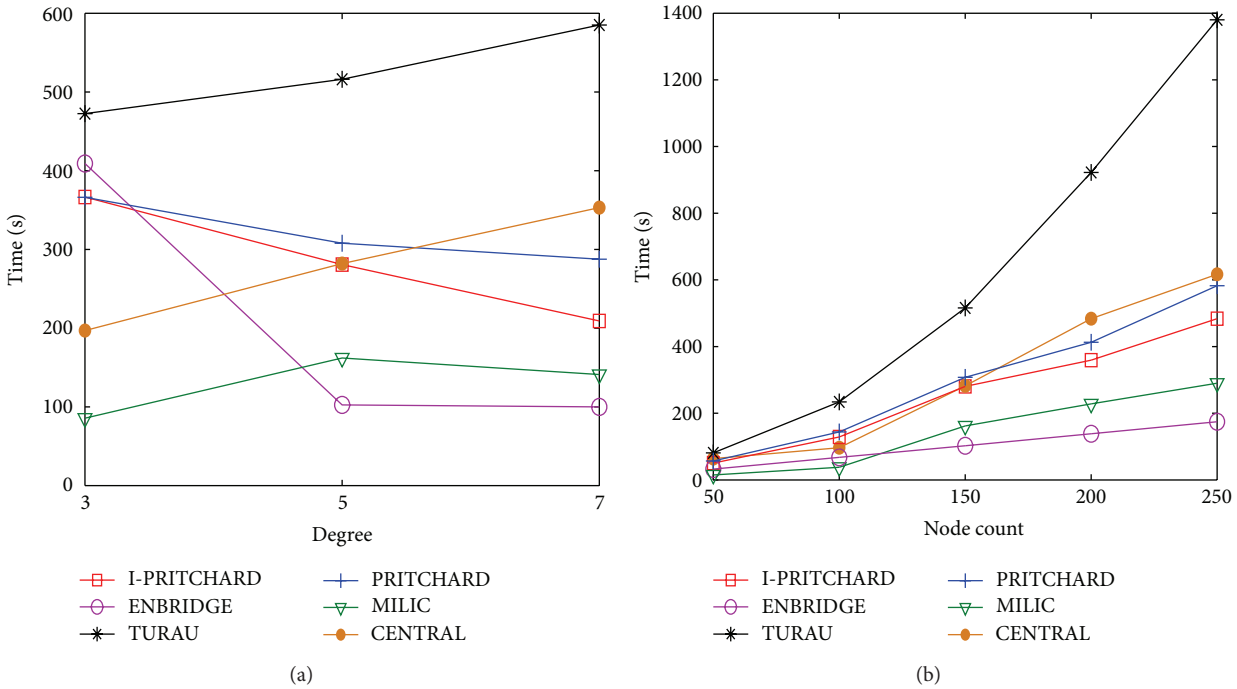


Figure 12: Wallclock times of algorithms against node degree. (b) Wallclock times of algorithms against node count.

consumes 1.7 times less energy than PRITCHARD algorithm. The energy consumption performance order of algorithms is ENBRIDGE, I-PRITCHARD, TURAU, PRITCHARD, MILIC, and CENTRAL. The energy consumption of ENBRIDGE is approximately 1.6 times better than TURAU, 2.6 times better than PRITCHARD, 4.3 times better than MILIC, and 6.5 times better than CENTRAL algorithm. This is a significant improvement over battery-powered sensor nodes in order to maximize the network lifetime.

6.4. Wallclock Times. Lastly, we measured the wallclock times of the distributed bridge detection algorithms. Firstly, we measured the wallclock times of I-PRITCHARD and ENBRIDGE algorithms against node count and node degree which are shown in Figures 11(a) and 11(b), respectively. Since the network diameter decreases when the degree increases, the wallclock times of both algorithms decrease. A sharp increase in the wallclock time of ENBRIDGE algorithm can be observed in Figure 11(b) when the node count is 150 and the degree is 3. The reason of this sharp increase is the fact that for especially sparse networks, ENBRIDGE runs both of the phases. Since the second phase is improved DFS based cut bridge detection, time consumption increases.

Wallclock times of I-PRITCHARD, ENBRIDGE, and their counterparts are given in Figures 12(a) and 12(a). The wallclock times of ENBRIDGE are the smallest, and those of I-PRITCHARD are the second smallest among the other algorithms. TURAU has the worst performance since it is DFS based and uses unicast for the message transmission. I-PRITCHARD is better than PRITCHARD in all cases since its phase count is 1 less. MILIC algorithm performs well for most of the cases. Although the performance of the ENBRIDGE is not good for some cases as explained in the previous paragraph, the algorithm performs best among other algorithms on the average since it completes its operation within a BFS session in most of the simulations.

In this section, we showed that our analytical results given in Section 5 conform with the simulations results that ENBRIDGE and I-PRITCHARD outperform the previously proposed approaches in terms of energy and time consumptions.

7. Conclusions

We proposed two distributed energy-efficient bridge detection algorithms I-PRITCHARD and ENBRIDGE. Our first algorithm, is the improved version of the Pritchard's algorithm. In this algorithm, we merged two phases into a single phase which leads to the removal of a downcast operation. Besides, we used radio broadcast communication instead of unicast message transfer that leads to the reduction of message header transmissions. The original idea of the second algorithm is to process proposed rules on 2-hop neighborhood information during a BFS session in order to detect bridges and classify all edges. With the help of these methods, our algorithms have $O(N)$ sent message complexity, $O(\Delta N)$ received and overheard message complexity where the largest message is $O(\Delta \log_2(N))$ bits.

We showed the detailed design of the proposed algorithms with example operations. We analyzed the proof of correctness, message complexity, time complexity, space complexity, and computational complexity and compared them with the previous work. We implemented the algorithm on TOSSIM environment with its counterparts. From our extensive simulations, we showed that the received and sent byte counts of I-PRITCHARD are always less than PRITCHARD. Besides, I-PRITCHARD consumes less energy and time than PRITCHARD in all simulation steps. These simulation results conform with the theoretical analysis and show that I-PRITCHARD improves PRITCHARD both theoretically and practically. Our second designed algorithm, ENBRIDGE, has the best simulation performance in terms of received byte count, sent byte count, energy consumption, and wallclock times. The energy savings of ENBRIDGE algorithm when compared to the other approaches are between 1.6 and 4.3. This is a significant improvement for energy-efficient bridge detection in sensor networks in order to detect weak points of the network and to prevent possible faults.

References

[1] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson, "Wireless sensor networks for habitat monitoring," in *Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications (WSNA '02)*, pp. 88–97, Atlanta, Ga, USA, September 2002.

[2] S. H. Lee, S. Lee, H. Song, and H. S. Lee, "Wireless sensor network design for tactical military applications: remote large-scale environments," in *Proceedings of the 28th IEEE Conference on Military Communications (MILCOM '09)*, pp. 911–917, Boston, Mass, USA, 2009.

[3] S. Hussain, S. Schaffner, and D. Moseychuck, "Applications ofwireless sensor networks and RFID in a smart home environment," in *Proceedings of the 7th Annual Communication Networks and Services Research Conference (CNSR '09)*, pp. 153–157, Moncton, Canada, May 2009.

[4] R. Tarjan, "Depth-first search and linear graph algorithms," *SIAM Journal on Computing*, vol. 1, pp. 146–160, 1972.

[5] Y. D. Liang and C. Rhee, "Optimal algorithm for finding biconnected components in permutation graphs," in *Proceedings of the ACM Computer Science Conference (CSC '95)*, pp. 104–108, Nashville, Tenn, USA, March 1995.

[6] R. Thurimella, "Sub-linear distributed algorithms for sparse certificates and biconnected components," in *Proceedings of the 14th Annual ACM Symposium on Principles of Distributed Computing (PODC '95)*, pp. 28–37, Ontario, Canada, August 1995.

[7] B. Milic, N. Milanovic, and M. Malek, "Prediction of partitioning in location-aware mobile ad hoc networks," in *Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS '05)*, vol. 9, p. 306, Big Island, Hawaii, USA, January 2005.

[8] V. Turau, "Computing bridges, articulations, and 2-connected components in wireless sensor networks," in *Proceedings of the 2nd international conference on Algorithmic Aspects of Wireless Sensor Networks (ALGOSENSORS '06)*, pp. 164–175, Springer, Venice, Italy, 2006.

[9] B. Milic and M. Malek, "Adaptation of the breadth first search algorithm for cut-edge detection in wireless multihop networks," in *Proceedings of the 10th ACM Symposium on Modeling, Analysis, and Simulation of Wireless and Mobile Systems (MSWiM '07)*, pp. 377–386, Chania, Greece, October 2007.

[10] S. Wang, X. Mao, S. J. Tang, X. Y. Li, J. Zhao, and G. Dai, "On movement-assisted connectivity restoration in wireless sensor and actor networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 4, pp. 687–694, 2011.

[11] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, McGraw-Hill, Cambridge, UK, 2nd edition, 2001.

[12] K. Lu, L. Huang, Y. Wan, and H. Xu, "Energy-efficient data gathering in large wireless sensor networks," in *Proceedings of the 2nd International Conference on Embedded Software and Systems (ICESS '05)*, pp. 327–331, Xi'an, China, 2005.

[13] H. Haanpaa, A. Schumacher, T. Thaler, and P. Orponen, "Distributed computation of maximum lifetime spanning subgraphs in sensor networks," in *Proceedings of the 3rd International Conference on Mobile Ad Hoc and Sensor Networks (MSN '07)*, pp. 445–456, Springer, Beijing, China, 2007.

[14] K. Erciyes, D. Ozsoyeller, and O. Dagdeviren, "Distributed algorithms to form cluster based spanning trees in wireless sensor networks," in *Proceedings of the 8th International Conference on Computational Science (ICCS '08)*, pp. 519–528, Springer, Kraków, Poland, 2008.

[15] S. Chen, M. Coolbeth, H. Dinh, Y.-A. Kim, and B. Wang, "Data collection with multiple sinks in wireless sensor networks," in *Proceedings of the 4th International Conference on Wireless Algorithms, Systems, and Applications (WASA '09)*, pp. 284–294, Springer, Boston, Mass, USA, 2009.

[16] V. Savic, A. Población, S. Zazo, and M. García, "An experimental study of RSS-based indoor localization using nonparametric belief propagation based on spanning trees," in *Proceedings of the 4th International Conference on Sensor Technologies and Applications (SENSORCOMM '10)*, pp. 238–243, Venice, Italy, July 2010.

[17] I. Cidon, "Yet another distributed depth-first-search algorithm," *Information Processing Letters*, vol. 26, no. 6, pp. 301–305, 1988.

[18] W. Hohberg, "How to find biconnected components in distributed networks," *Journal of Parallel and Distributed Computing*, vol. 9, no. 4, pp. 374–386, 1990.

[19] P. Chaudhuri, "An optimal distributed algorithm for computing bridge-connected components," *Computer Journal*, vol. 40, pp. 200–207, 1997.

[20] Y. H. Tsin, "Some remarks on distributed depth-first search," *Information Processing Letters*, vol. 82, pp. 173–178, 2002.

[21] D. Pritchard, "An optimal distributed bridge-finding algorithm," in *Proceedings of the 25th Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC '06)*, Denver, Colo, USA, July 2006.

[22] O. Dagdeviren and K. Erciyes, "Graph matching-based distributed clustering and backbone formation algorithms for sensor networks," *Computer Journal*, vol. 53, no. 10, pp. 1553–1575, 2010.

[23] P. Sommer and R. Wattenhofer, "Gradient clock synchronization in wireless sensor networks," in *Proceedings of the International Conference on Information Processing in Sensor Networks (IPSN '09)*, pp. 37–48, San Francisco, Calif, USA, April 2009.

[24] H. Cheng, Q. Liu, and X. Jia, "Heuristic algorithms for real-time data aggregation in wireless sensor networks," in *Proceedings of the International Wireless Communications and Mobile Computing Conference (IWCMC '06)*, pp. 1123–1128, Vancouver, Canada, July 2006.

[25] "IEEE Standard for Information technology-Telecom. and information exchange between systems-Local and metropolitan area networks-Specific requirements-Part 15.4: Wireless Medium Access Control and Physical Layer Specifications for Low-Rate Wireless Personal Area Networks," 2003.

[26] "IEEE 802.11 Standard Working Group, Draft Standard for Information Technology—Telecom. and Information Exchange Between Systems—LAN/MAN Specific requirements Part 11: Wireless LAN Medium Access Control and Physical Layer Specifications," 2007.

[27] W. Ye, J. Heidemann, and D. Estrin, "Medium access control with coordinated adaptive sleeping for wireless sensor networks," *IEEE/ACM Transactions on Networking*, vol. 12, no. 3, pp. 493–506, 2004.

[28] L. Choi, S. H. Lee, and H. Choi, "M-mac: mobility-based link management protocol for mobile sensor networks," in *Proceedings of the Software Technologies for Future Dependable Distributed Systems (STF-SSD '09)*, pp. 210–214, Tokyo, Japan, 2009.

[29] J. Polastre, J. Hill, and D. Culler, "Versatile low power media access for wireless sensor networks," in *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys '04)*, pp. 95–107, Baltimore, Md, USA, 2004.

[30] V. Rajendran, K. Obraczka, and J. J. Garcia-Luna-Aceves, "Energy-efficient, collision-free medium access control for wireless sensor networks," *Wireless Networks*, vol. 12, no. 1, pp. 63–78, 2006.

[31] B. Awerbuch, "A new distributed depth-first-search algorithm," *Information Processing Letters*, vol. 20, pp. 147–150, 1985.

[32] K. B. Lakshmanan, N. Meenakshi, and K. Thulasiraman, "A time-optimal message-efficient distributed algorithm for depth-first-search," *Information Processing Letters*, vol. 25, no. 2, pp. 103–109, 1987.

[33] M. B. Sharma and S. S. Iyengar, "An efficient distributed depth-first-search algorithm," *Information Processing Letters*, vol. 32, pp. 183–186, 1989.

[34] T.-Y. Cheung, "Graph traversal techniques and the maximum flow problem in distributed computation," *IEEE Transactions on Software Engineering*, vol. 9, pp. 504–512, 1983.

[35] D. Kumar, S. S. Iyengar, and M. B. Sharma, "Corrigenda: corrections to a distributed depth-first search algorithm," *Information Processing Letters*, vol. 35, pp. 55–56, 1990.

[36] S. A. M. Makki and G. Havas, "Distributed algorithms for depth-first search," *Information Processing Letters*, vol. 60, no. 1, pp. 7–12, 1996.

[37] P. Levis, N. Lee, M. Welsh, and D. Culler, "TOSSIM: accurate and scalable simulation of entire TinyOS applications," in *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems (SenSys '03)*, pp. 126–137, Los Angeles, Calif, USA, November 2003.

Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2013, Article ID 527965, 14 pages
http://dx.doi.org/10.1155/2013/527965

Research Article

Node Selection Algorithms with Data Accuracy Guarantee in Service-Oriented Wireless Sensor Networks

Hongju Cheng, Ronglie Guo, and Yuzhong Chen

College of Mathematics and Computer Science, Fuzhou University, Fuzhou 350108, China

Correspondence should be addressed to Hongju Cheng; cscheng@fzu.edu.cn

Received 20 December 2012; Accepted 26 February 2013

Academic Editor: Neal N. Xiong

The service-oriented architecture is considered as a new emerging trend for the future of wireless sensor networks in which different types of sensors can be deployed in the same area to support various service requirements. The accuracy of the sensed data is one of the key criterions because it is generally a noisy version of the physical phenomenon. In this paper, we study the node selection problem with data accuracy guarantee in service-oriented wireless sensor networks. We exploit the spatial correlation between the service data and aim at selecting minimum number of nodes to provide services with data accuracy guaranteed. Firstly, we have formulated this problem into an integer nonlinear programming problem to illustrate its NP-hard property. Secondarily, we have proposed two heuristic algorithms, namely, Separate Selection Algorithm (SSA) and Combined Selection Algorithm (CSA). The SSA is designed to select nodes for each service in a separate way, and the CSA is designed to select nodes according to their contribution increment. Finally, we compare the performance of the proposed algorithms with extended simulations. The results show that CSA has better performance compared with SSA.

1. Introduction

Sensing is considered as one of the most important technologies especially in the emerging big data era. Nodes with sensing ability can be deployed everywhere in the world including airspace, ground, and underwater environment due to their cheapness, simplicity, and small size. Moreover, the wireless radio allows these sensor nodes to be organized into a network, which is generally named as Wireless Sensor Networks (WSN) [1], and local information about the environment is then sensed and reported to the base station in a periodical manner. It is obvious that the wireless sensor networks will create a huge number of data with time ongoing and the number of network increasing. Accordingly, one challenging issue is how to utilize these various wireless sensor networks in the future big data era.

The current wireless sensor networks are generally data-centric or application-centric, which means that each sensor node serves for one special application. However, with the number of different applications increasing rapidly, heterogeneous wireless sensor networks appear and they might be located in the same physical areas providing different data-collection functions. How to connect these heterogeneous wireless sensor networks efficiently is still a pioneering work in the future ubiquitous computing environment due to several observations. Firstly, two different applications may be interested in the same collected data, and it is unnecessary to place two separate nodes with identical sensing devices but different tasks. Secondary, in case that one application is concerned with several different types of sensed data simultaneously, such a requirement is not guaranteed since current solutions work only in a separate way. Finally, the emergence of powerful sensors, which can provide different types of data sensing, has introduced new issues in the research of wireless sensor networks since they can support different applications simultaneously.

Accordingly, the service-oriented architecture appears as a new emerging trend for the future of wireless sensor networks, in which sensor networks are considered as the services provider and sensors as the data sources for these services [2]. Users and programmers can access the service-oriented wireless sensor networks by using a simple

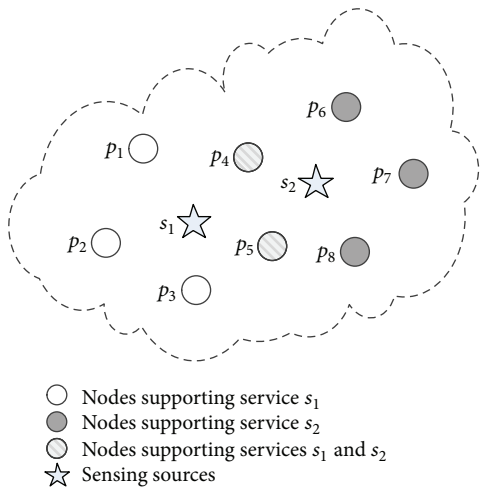


Figure 1: An example of a service-oriented wireless sensor network in which service s_1 is supported by nodes p_1, p_2, p_3, p_4 and p_5, service s_2 by nodes p_4, p_5, p_6, p_7 and p_8.

service-oriented interface and utilizing encapsulation of the low-level implement details. Services in wireless sensor networks may be the sensing capabilities for example, temperature and humidity or software components provided by nodes, for example, the operations of in-network data aggregation, time synchronization, and data processing [2–5]. The data sensing of nodes can also be defined as the sensing services, and sensed data as the service data similarly. Furthermore, the sensor nodes can be equipped with multiple types of sensing units used to collect environmental information. For example, the MICA2 mote [6] can provide services such as light, sound, and vibration.

In the heterogeneous applications scenarios, different types of sensors can be deployed in the same area to support various service requirements. Figure 1 shows an example of a service-oriented wireless sensor network including eight sensor nodes. The sensed sources are assumed to be located at positions s_1 and s_2, and each sensed target is assumed one different service. Nodes p_1, p_2, and p_3 can only support service s_1, and nodes p_6, p_7, and p_8 support service s_2. However, nodes p_4 and p_5 can support service s_1 and s_2 simultaneously. In this example, there are several ways to select nodes while providing the services s_1 and s_2 by assuming that at least two nodes are required for each service, that is, nodes p_1, p_2 for service s_1 and nodes p_6, p_7 for s_2, or nodes p_4, p_5 for both of them simultaneously. Note that sensor networks are generally deployed in dense manner and a huge number of nodes can be provided to collect data from the environment. It leads to the problem of how to select nodes in an efficient way to provide the required services.

The accuracy of the sensed data is one of the key criterions while choosing nodes to provide required service. It is also known that sensor nodes are generally designed under the guideline of cheapness and simplicity and that they are mostly deployed in a dynamic and rough terrain for continuous environmental monitoring. The sensing equipment on sensor nodes is expected to be unreliable and the collected data to be distorted. Furthermore, some physical attributes exhibit a gradual and continuous variation over the two-dimensional Euclidean space due to the diffusion property and, thus, each observer has different distorted data. To collect all related data around the same sensing sources can help to eliminate or minimize the distortion. However, it might lead to heavy energy consumption in the network. Some applications only are concerned with approximate observations rather than exact results [7, 8], and it is unnecessary to gather data from all nodes in the network in this case. It shall also be mentioned that the diffusion property of physical attributes results in spatial correlation among sensed data observed by nodes closer to the same sensing source. Exploiting the spatial correlation can help to improve the network performance by selecting a subset of nodes to provide the required service with data accuracy guaranteed [9, 10].

Although the service-oriented architecture for the wireless sensor networks has already been introduced recently in related works [4, 5, 11–14], most of them are concerned with the practical architecture framework, such as middleware and platforms. Some works [13, 14] considered the node scheduling to support services query with network lifetime prolonged, and some others are concerned with the spatial correlationship among the sensed data with the service unconsidered [15–17]. To provide accurate service to users is an important issue and it is generally one criterion for the applications. In this paper, we focus on the heterogeneous service supporting problem in the future wireless sensor networks and aim at providing efficient node selection algorithms with data accuracy guarantee for the service-oriented sensor networks. Different from previous works, we consider the data accuracy for services by following the observation that sensed data is a noisy version of the physical phenomenon. And we have explored the data accuracy according to their spatial correlation for the same sensing sources.

Due to the inaccuracy and spatial correlation of the sensed data, it is a new and challenging issue to provide the required services with data accuracy guarantee in an *energy-efficient* way for the service-oriented wireless sensor networks. So far, as we know, this is the first paper concerned with both the data inaccuracy and spatial correlation in the sensor networks, and we aim at providing node selection algorithms so as to improve the network performance. The main contributions of this paper are summarized as follows. (1) We have proposed the node selection problem with data accuracy guarantee for service-oriented wireless sensor networks via bipartite graph and formulated it as an integer nonlinear programming problem to illustrate its NP-hard property. (2) We have also presented two efficient heuristic algorithms for this problem; namely, Separate Selection Algorithm (SSA) and Combined Selection Algorithm (CSA) with low-time complexity.

The rest of this paper is organized as follows. In Section 2, we summarize the related works. Section 3 describes the system model and the problem formulation. Sections 4 and 5 have introduced the integer nonlinear programming formulation and two heuristic selection algorithms. Section 6

describes and analyzes the simulation results. We conclude this paper in Section 7.

2. Related Works

This paper focuses on the node selection problem in service-oriented wireless sensor networks. Several works have been done to develop service-oriented architecture specific to the sensor networks, and the architecture has shown many advantages in the heterogeneous applications scenarios. Gračanin et al. [2] proposed a service-centric model that focuses on services provided by a wireless sensor networks and views a wireless sensor networks as a service provider. This model consists of mission, network, region, sensor, and capability layers. Within each layer, there are four planes or functionality sets: communication, management, application, and generational learning. Rezgui and Eltoweissy [4] introduced the service-oriented architecture as an approach for building a new generation of open, efficient, interoperable, scalable, application-aware sensor-actuator networks. In this vision, sensor-actuator networks would not be deployed to provide sensing and actuation capabilities to a specific application but, rather, to provide sensing and actuation services to any application. King et al. [5] developed a service-oriented sensor and actuator platform called Atlas, which enables self-integrative, programmable pervasive spaces. This platform has shown the advantage of improving communication and interoperability between heterogeneous devices in pervasive computing environments. Authors of [11] proposed TinySOA, a service-oriented architecture that allows programmers to access wireless sensor networks from their applications by using a simple service-oriented API via the language of their choice. The main advantage of TinySOA is relieving application developers from dealing with the low-level technical details of the wireless sensor networks to get sensors data. Corchado et al. [12] proposed a service-oriented telemonitoring system for healthcare using heterogeneous wireless sensor networks, which aimed at improving healthcare and assistance for dependent people.

It is an important issue to provide services efficiently with resource-constrained sensor nodes. Node scheduling is considered as an efficient technique to implement the service-supporting schemes, in which sensor nodes should be selected to provide requested services. Recently, Wang et al. [13] investigated the service-availability-aware sleep scheduling design in service-oriented wireless sensor networks. The purpose of this study is to minimize the energy consumption and guarantee that enough sensors are active to ensure service availability at all times. The authors had proven this problem to be NP-hard and presented heuristic linear-programming based-solutions. However, they assumed that each service has a known requirement on the number of active sensors based on the historical service composition requests in the system, which may not be the case in practice. Furthermore, they only consider the sleep scheduling design for the sensors in the service provider overlay network and neglect the routing cost of service data. Authors of [14] try to identify the service composition that is less likely to be invalid in the near future due to nodes going to sleep mode. The goal is to minimize the recomposition cost. They make use of the dynamic programming to reduce total service composition cost when the minimum number of required service composition solutions is derived. However, the dynamic programming is unsuitable for large-scale problems.

The distributed nature of wireless sensor network results in spatial correlation among the sensed data. And data accuracy is accordingly influenced by the spatial correlationship. Under different assumptions, researchers have proposed several mathematical models for spatial correlation in wireless sensor networks. Some [18] assume that the sensed data follow diffusion property, and some [17] use an empirically obtained approximation function for the joint entropy of sensed data. The most commonly used model is the jointly Gaussian [9, 19, 20], which assumes the data to be jointly Gaussian with the correlation being a function of the distance. The jointly Gaussian model is easy to use and analyze. However, the chief limitation is that it forces the joint probability density function of the data values to be jointly Gaussian. Some researchers [21] use variograms to analyze spatial correlation in wireless sensor networks. The proposed model is Markovian in nature and can capture correlation in data irrespective of the node density, the number of source nodes, or the topology. Furthermore, this model derives the data value at a node from other correlated nodes whose data values have already been derived. However, it is not always the case that a given spatial process will be Markovian. Some others proposed correlation model for specific applications, such as soil moisture measurement in wireless underground sensor networks [10]. The presence of spatial correlation among sensor network data has been exploited for solving different problems. The authors in [15] proposed a traffic model for wireless sensor networks, which takes into account the statistical patterns of node mobility and spatial correlation. In [16, 17], spatial correlation was used to design energy-efficient data aggregation algorithms. Ma et al. [16] proposed a distributed clustering algorithm based on the dominating set theory to choose the cluster heads nodes and construct clusters by measuring the spatial correlation between sensors. Pattem et al. [17] studied the correlated data gathering problem and followed the idea of using an empirically obtained approximation function for the joint entropy of sources.

It is important and challenging to provide different services with data accuracy guaranteed through unreliable sensors nodes. Fault tolerance is one of the most important techniques, which has been taken into consideration in many works [22–27]. Han et al. [22] addressed the problem of deploying minimum number of relay nodes to achieve diverse levels of fault tolerance with higher network connectivity in the context of heterogeneous wireless sensor networks. However, they adopted the network model that in which nodes possess different transmission radius, while all of the relay nodes use an identical transmission radius. Banerjee et al. [23] investigated the event detection scheme with fault tolerance for multiple events occurring simultaneously. They proposed the use of polynomial-based scheme that addresses the problems of event region detection by having an aggregation tree of sensor nodes. However, their

work is limited to static sensor and the network topology cannot adapt to the dynamic nature of simultaneous events with varied priorities.

Our work in this paper is concerned with the node selection algorithms, which is similar to the works that aim at dealing with node selection and assignment problems. Cai et al. [28] addressed the multiple directional cover sets problem of organizing the directions of sensors into a group of nondisjoint cover sets to extend the network lifetime. The directional sensors are different from common sensors that have a limited angle of sensing range. The authors proved this problem is NP-complete and presented three heuristic approaches. Lin et al. [29] proposed an adaptive energy-efficient multisensor scheduling scheme for collaborative target tracking in wireless sensor networks. The challenging issue of this problem is how to achieve energy efficiency and track reliability while satisfying the tracking accuracy requirement. In their algorithm, a number of sensors are selected to form a temporary tasking cluster, and the optimal sampling interval is determined to satisfy the given tracking accuracy. Johnson et al. [30] considered sensor-mission assignment problem in wireless sensor networks. In this problem, multiple missions compete for sensor resources. They showed that this problem is NP-hard even to approximate, and presented several heuristic algorithms. Liu et al. [31] studied the topology control problem using a probabilistic network model. They attempted to find a minimal transmission range for each node while the global network reachability satisfies certain threshold. Different from these previous works, we aim at providing efficient node selection algorithms with data accuracy guaranteed for the service-oriented wireless sensor networks by exploring the spatial correlation among the sensed data and the advantages of diverse services provided by different sensor nodes.

3. System Model and Problem Formulation

In this section, we have firstly introduced the network model for the service-oriented wireless sensor networks. Secondly, we have described the spatial correlation model for single as well as multiple services in the network. Finally, we have formulated a definition for the node selection problem with data accuracy guaranteed in the service-oriented wireless sensor networks.

3.1. Network Model. We consider a wireless sensor network in the plane with stationary nodes $P = \{p_1, p_2, \ldots, p_n\}$, which are built to provide a series of services $S = \{s_1, s_2, \ldots, s_m\}$. Each node p_i can provide one or more service S_i, which is a subset of S; that is, $S_i \subseteq S$. One service, for example, s_j, can be provided by a group of nodes P_j, and $P_j = \{p_i \mid s_j \in S_i\}$. It is obvious that the set size $|P_j|$ demonstrates the number of nodes in the network which can provide service s_j. The relationship between nodes and services can be further described as a bipartite graph $G = (P, S, E)$, where P denotes the set of nodes, S denotes the set of services, and E denotes the set of edges. There is an edge (p_i, s_j) between p_i and s_j in case that p_i can provide service s_j; that is, $(p_i, s_j) \in E$. Figure 2 has shown an example of the proposed model for service-oriented sensor network with five nodes and each service is supported by three distinct nodes.

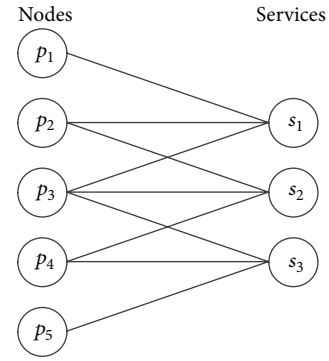


Figure 2: An example of the bipartite graph in which each service is supported by three distinct nodes.

To be convenient, the symbols used in this work are summarized in Table 1.

3.2. Spatial Correlation Model. Researchers have proposed several mathematical models for spatial correlation in wireless sensor networks under different assumptions. Pattem et al. [17] proposed to use an empirically obtained approximation function for the joint entropy of sensed data. In [18], the sensed data is assumed to follow the diffusion property and the diffusion is formulated as a function of the distance. The jointly Gaussian is adopted in many related works [9, 19, 20], which assumes the data to be jointly Gaussian with the correlation as a function of the distance. The jointly Gaussian model is easy to use and analyze by forcing the joint probability density function of the data values to be jointly Gaussian. Jindal and Psounis [21] analyzed the spatial correlation among sensed data by using variograms in wireless sensor networks. The proposed model is a special case of Markov random field. In this model, the data value at a node is derived from other correlated nodes whose data values have already been derived. However, it is not always the fact that a given spatial process will be Markovian. In this paper, we are concerned with the data accuracy with the spatial correlation model. The jointly Gaussian model proposed in [9] has considered the measurement noise of nodes and given the distortion function, which is suitable for our problem.

In the senor networks, the observation result of each node is in fact a noisy version of the physical phenomenon located at the sensing source, and it can be modeled as Gaussian random variable V of zero mean and variable σ_V^2; that is, $V \sim \mathcal{N}(0, \sigma_V^2)$. Similarly, the sensed data for the physical phenomenon at node p_i can also be modeled as Gaussian variable V_i, $V_i \sim \mathcal{N}(0, \sigma_V^2)$. Assume that the sensed data for node p_i/p_j is denoted as V_i/V_j accordingly. The correlations between V and V_i, V_i and V_j are described as

$$\begin{aligned} \operatorname{corr}\{V, V_i\} &= \rho_{v,i} = K_\vartheta\left(d_{v,i}\right), \\ \operatorname{corr}\left\{V_i, V_j\right\} &= \rho_{i,j} = K_\vartheta\left(d_{i,j}\right), \end{aligned} \tag{1}$$

where $d_{v,i}$ denotes the Euclidean distance between p_i and the sensing source V, $d_{i,j}$ denotes the Euclidean distance between p_i and p_j, $K_\vartheta(\cdot)$ is the covariance function concerned with the Euclidian distance d and it is formulated as

$$K_\vartheta(d) = e^{-(d/\theta)}, \quad \theta > 0, \tag{2}$$

where θ controls the correlation between the distances of sensor nodes. In addition, we can see that $K_\vartheta(\cdot) = 1$ in case that $d = 0$, and $K_\vartheta(\cdot) = 0$ in case that $d = +\infty$.

The collected data by the sensor node p_i is often subject to noise interference originated from the environment, and it can be represented as

$$X_i = V_i + N_i, \tag{3}$$

where N_i is the additive white Gaussian noise, $N_i \sim \mathcal{N}(0, \sigma_N^2)$. We assume that the noise that each sensor node encounters is independent of each other.

According to [9], the distortion of the estimation for V is formulated as

$$\begin{aligned}\widehat{\delta} &= \sigma_V^2 - \frac{\sigma_V^4}{L(\sigma_V^2 + \sigma_N^2)}\left(2\sum_{i=1}^{L}\rho_{v,i} - 1\right) \\ &\quad + \frac{\sigma_V^6}{L^2(\sigma_V^2 + \sigma_N^2)^2}\sum_{i=1}^{L}\sum_{j\neq i}^{L}\rho_{i,j},\end{aligned} \tag{4}$$

where L ($L > 0$) is the number of sensor nodes.

We use σ_V^2 to normalize $\widehat{\delta}$, and the estimated data accuracy is calculated as

$$\begin{aligned}\widehat{a} &= 1 - \frac{\widehat{\delta}}{\sigma_V^2} \\ &= \frac{\beta}{L(\beta+1)}\left(2\sum_{i=1}^{L}\rho_{v,i} - 1\right) \\ &\quad - \frac{\beta^2}{L^2(\beta+1)^2}\sum_{i=1}^{L}\sum_{j\neq i}^{L}\rho_{i,j},\end{aligned} \tag{5}$$

where $\beta = \sigma_V^2/\sigma_N^2$ denotes the Signal-to-Noise Ratio (SNR).

3.3. Problem Definition. In this paper, we study the problem of node selection in the service-oriented wireless sensor networks with the data accuracy requirement guaranteed for the services. The number of nodes is considered as the optimizing object due to the following considerations. Firstly, there are fewer packets to be transmitted in the network if we select less number of nodes to provide services, which is also helpful to reduce the energy consumption. Secondarily, it will increase the collision probability in the contention-based wireless network if too many nodes are kept awake, and significant retransmission cost and additional delay occur accordingly. Finally, it helps to reduce the overhead of data transmission to allow one node to provide multiple services simultaneously. In case that the data of different services is correlated, it can be compressed into a smaller packet; even in the uncorrelated case, it can still be transmitted in a single packet, and thereby it is helpful to reduce overhead in the network [32].

In this paper, we aim at providing node strategies with the number of selected nodes minimized. Let a_j be the data accuracy requirement of each service s_j, P_j' one subset of nodes selected from P_j to provide service s_j, $P_j' \subseteq P_j$, and $\widehat{a}_j(P_j')$ the estimated data accuracy of service s_j when service s_j is provided by nodes in set P_j'. The node selection problem for the service-oriented wireless sensor networks can be defined as follows: given a bipartite graph $G = (P, S, E)$, in which P denotes the set of nodes, S denotes the set of services, and E denotes the set of edges between P and S, and given the required data accuracy requirement a_j of each service $s_j \in S$ and spatial correlation among these nodes and corresponding sensing sources, the problem is to find a subgraph $G' = (P', S, E')$, $P' \subseteq P$, $E' \subseteq E$, and the objective is to minimize the number of selected nodes in P' under the constraint that $\widehat{a}_j(P_j') \geq a_j$ is satisfied for each service $s_j \in S$.

4. Integer Nonlinear Programming Formulation

In this section, we present an Integer Nonlinear Programming (INLP) formulation for the node selection problem. Integer programming is a mathematical optimization or a feasibility program in which some or all of the variables are restricted to be integers. INLP is a special case of integer programming, where some of the constraints or the objective functions are nonlinear. INLP is considered as an efficient technique to solve the optimization problem with nonlinear constraint, so that it is feasible to express the node selection problem as INLP. This paper is concernd with the node selection problem, and the objective is to minimize the total number of selected nodes with nonlinear data accuracy constraint. We use the following set of binary integer (0 or 1) variables and constraints in the INLP formulation.

(1) Variables $e_{i,j}$ for each node $p_i \in P$ and service $s_j \in S$. The variable $e_{i,j}$ is 1 if and only if $(p_i, s_j) \in E$; that is, node p_i can provide service s_j:

$$e_{i,j} = \begin{cases} 1, & \text{if node } p_i \text{ can provide service } s_j, \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

The variables $e_{i,j}$ can be obtained when the topology graph and the set of services provided by nodes are given. Obviously, the selected nodes for services s_j shall be selected from these nodes with $e_{i,j} = 1$.

(2) Variables $x_{i,j}$ for each node $p_i \in P$ and service $s_j \in S$. The variable $x_{i,j}$ is 1 if and only if node p_i is assigned to provide service s_j:

$$x_{i,j} = \begin{cases} 1, & \text{if node } p_i \text{ is assigned to provide service } s_j, \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

As we can see, the set $\{x_{i,j} \mid i = 1, 2, \ldots, n,\ j = 1, 2, \ldots, m.\}$ had indicated one scheduling scheme for the given wireless

sensor networks, in which each node p_i is assigned to provide service s_j if $x_{i,j} = 1$. Note that $x_{i,j}$ equals 0 in case that $e_{i,j} = 0$, which means that node p_i cannot be assigned to provide service s_j since it is not supported. In this way, we have the following constraint:

$$e_{i,j} - x_{i,j} \geq 0, \quad \forall i = 1, 2 \ldots, n, \ j = 1, 2 \ldots, m. \tag{8}$$

Let $\widehat{a}_j$ be the estimated data accuracy of service s_j, and $\widehat{a}_j$ can be obtained via formula (5), which can be further described as follows:

$$\widehat{a}_j = \begin{cases} \dfrac{\beta}{L_j(\beta+1)}\left(2\displaystyle\sum_{i=1}^{n} x_{i,j}\rho_{v,i} - 1\right) \\ \quad -\dfrac{\beta^2}{{L_j}^2(\beta+1)^2}\displaystyle\sum_{i=1}^{n}\sum_{k\neq i}^{n} x_{i,j}x_{k,j}\rho_{i,k}, & \text{if } L_j \neq 0 \\ 0, & \text{if } L_j = 0, \end{cases} \tag{9}$$

where $L_j = \sum_{i=1}^{n} x_{i,j}$ and L_j denotes the number of selected nodes which are assigned to provide service s_j.

In order to satisfy the required data accuracy for all services in the network, we have the following constraint:

$$\widehat{a}_j \geq a_j, \quad \forall j = 1, 2 \ldots, m. \tag{10}$$

(3) Variables y_i. The variable y_i is 1 if and only if node p_i is selected to provide services required in the network:

$$y_i = \begin{cases} 0, & \text{if node } p_i \text{ is not selected to provide any services,} \\ 1, & \text{otherwise.} \end{cases} \tag{11}$$

By following the above definition, y_i equals 0 if node p_i is not selected to provide any service, and otherwise it equals 1. As mentioned above, the variable $x_{i,j}$ denotes the case that node $p_i \in P$ can provide service $s_j \in S$ or not, and we have the following constraint:

$$my_i \geq \sum_{j=1}^{m} x_{i,j} \geq y_i, \quad \forall i = 1, 2 \ldots, n. \tag{12}$$

The objective of node selection problem is to minimize the total number of nodes that are selected to provide the required services, and the number of selected nodes can be calculated as $\sum_{i=1}^{n} y_i$.

Then the node selection problem discussed in this paper can be formulated as

$$\text{Minimized} \quad \sum_{i=1}^{n} y_i$$

Subject to

$$\begin{aligned} & x_{i,j} \in \{0, 1\}, \quad \forall i = 1, 2 \ldots, n, \ j = 1, 2 \ldots, m, \\ & y_i \in \{0, 1\}, \quad \forall i = 1, 2 \ldots, n, \\ & \widehat{a}_j \geq a_j, \quad \forall j = 1, 2 \ldots, m, \\ & e_{i,j} - x_{i,j} \geq 0, \quad \forall i = 1, 2 \ldots, n, \ j = 1, 2 \ldots, m, \\ & my_i \geq \sum_{j=1}^{m} x_{i,j} \geq y_i, \quad \forall i = 1, 2 \ldots, n. \end{aligned} \tag{13}$$

In this section, we have introduced an Integer Nonlinear Programming (INLP) formulation for the node selection problem. The proposed INLP is generally considered as an efficient way to provide an accurate description on the problem formulation. This formulation is useful to find the optimal solution in case that the solution space is small enough with the help of some well-known tools, such as LINGO and MATLAB. However, INLP is a special case of integer programming which is proved to be NP-hard, and accordingly the INLP problem is NP-hard. Many related works have also been proposed to find the suboptimal solution for a given INLP problem [33–35]. Although INLP has shown its good performance in the practical applications, it results in some well-known deflects such as computation as well as space complexity, especially in case that the solution space is very large. Unfortunately, the wireless sensor network generally includes hundreds to thousands of nodes; the variables required by the INLP mentioned above might increase exponentially with the node number. Another problem is that for a random network, it is hard to gather all the constraints mentioned above since these nodes are heterogonous and constraints for each node are fully different from each other. Moreover, in the practical applications, the sink is almost impossible to get the *accurate* information in the harsh environment by following the observation that the sensed data is generally a noisy version of the physical phenomenon. It is more practical to adopt a suboptimal solution with the data accuracy guaranteed instead of optimal one that is hard to find for an NP-hard problem. In this way, it is reasonable and necessary to develop heuristic algorithms for the node selection problem in the service-oriented wireless sensor networks.

5. Heuristic Algorithms

Heuristic algorithms are generally considered as an important way to solve the NP-hard problem. In this section, we propose two heuristic algorithms for the node selection problem in the service-oriented wireless sensor networks,

namely, the Separate Selection Algorithm (SSA) and the Combined Selection Algorithm (CSA).

5.1. Separate Selection Algorithm (SSA). The basic idea of the Separate Selection Algorithm (SSA) is that we select a minimum number of nodes for each required service with the data accuracy guaranteed in a separate way, and the union of selected nodes for all services is considered as the problem solution. The key process for the SSA is how to select nodes for each service. Here we follow the idea with which nodes are selected in a sequence way, and in each step, we will choose the node that is potential to improve the data accuracy.

Assume that the current node selection solution is $G' = (P', S, E')$, in which $P' = \{P'_j \mid j = 1, 2, \ldots, m\}$, $S = \{s_j \mid j = 1, 2, \ldots, m\}$ is the set of services, and $E' = \{(p_i, s_j) \mid p_i \in P', s_j \in S\}$. Let us consider the general case that one node, that is, p_i is considered to provide one special service, that is, s_j. Here we use $I_{i,j}$ to denote the data accuracy increment for service s_j in case that node p_i is selected to provide service s_j, where $(p_i, s_j) \in E$. $I_{i,j}$ can be calculated as follows.

(1) In case that $\widehat{a}_j(P'_j) \geq a_j$, we have $I_{i,j} = 0$, which means that the data accuracy requirement for service s_j has already been satisfied that there is no more improvement once nodes p_i and (p_i, s_j) are added into the final solution.

(2) In case that $\widehat{a}_j(P'_j + \{p_i\}) \leq \widehat{a}_j(P'_j)$, we have $I_{i,j} = 0$, which means that the data accuracy of service s_j cannot be increased once nodes p_i and (p_i, s_j) are added into the final solution.

(3) In case that $\widehat{a}_j(P'_j) < \widehat{a}_j(P'_j + \{p_i\}) \leq a_j$, we have $I_{i,j} = \widehat{a}_j(P'_j + \{p_i\}) - \widehat{a}_j(P'_j)$, which denotes the increment of the data accuracy for service s_j in case that nodes p_i and (p_i, s_j) are added into the final solution.

(4) In case that $\widehat{a}_j(P'_j) < a_j < \widehat{a}_j(P'_j + \{p_i\})$, we have $I_{i,j} = a_j - \widehat{a}_j(P'_j)$, which means that we generally neglect the part of increment that exceeds the requirement since the solution is required only to provide the asked data accuracy.

The pseudocode for SSA is listed in Table 1. In Line 1, the set of selected nodes P'_j for each service s_j is initially set as $\emptyset$. In Lines 2–9, the algorithm tries to select nodes for each service $s_j \in S$. In case that the current data accuracy cannot satisfy the data accuracy requirement, that is, $\widehat{a}_j(P'_j) < a_j$, we firstly check all the candidate nodes that are useful to improve data accuracy. Secondarily, we select one node with maximum data accuracy increment as the candidate (in Line 4). Finally, the selected node is added into P'_j to provide service s_j (in Line 6). This process continues until enough nodes are selected for all services.

We illustrate the SSA algorithm by an example given in Figure 3. As we can see from Figure 3(a), the network has four nodes and each service is supported by three distinct nodes. The required data accuracy for each service is listed on the right side; for example, the required data accuracy of s_1 is 0.8,

Table 1: Description of the symbols.

Symbol	Description
n	The number of nodes
m	The number of services
G	The node-service bipartite graph
P	The set of nodes
S	The set of services
E	The set of edge between P and S
G'	The node-service assignment bipartite graph
P'	The set of nodes that selected to provide services
E'	The set of edge between P' and S
P_j	The set of nodes that can provide service s_j
P'_j	The subset of nodes that selected from P_j to provide service s_j
S_i	The set of services that node p_i can provide
a_j	The data accuracy requirement of service s_j
$\widehat{a}_j(A)$	The estimated data accuracy of service s_j when service s_j is provide by nodes in set A

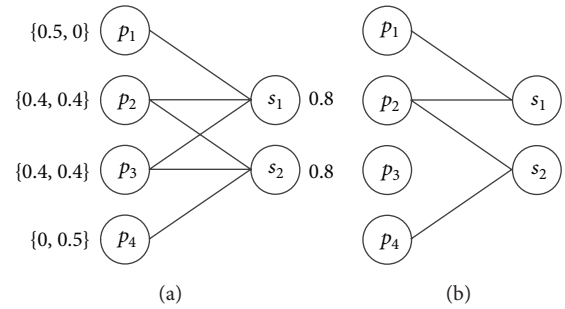


Figure 3: An example of the execution of SSA algorithm. (a) The node-service bipartite graph $G = (P, S, E)$. (b) The final solution of node-service selection bipartite graph $G' = (P', S, E')$.

and the data accuracy increment of each node when services is provided only by this node is also listed on the left side; for example, the data accuracy increment of p_1 is 0.5 for s_1 and 0.0 for s_2. Without loss of generality, we assume that the data accuracy increment of each node can be added directly to simplify description; that is, the data accuracy of s_1 provided by p_1 and p_2 is 0.5 + 0.4 that equals 0.9. The algorithm will select nodes for s_1 and then s_2. Node p_1 with the maximum data accuracy increment for s_1 will be selected firstly. In case that the algorithm has selected p_1, both node p_2 and p_3 can guarantee the required data accuracy of s_1, and we assume that the randomly selected one is p_2. Similarly, service s_2 will firstly select p_4. In case that the algorithm has selected p_4, both p_2 and p_3 can guarantee the required data accuracy of s_2. However, p_2 is intended to be selected by s_2 in case that it is already selected by s_1. After that, the selected nodes have guaranteed the required data accuracy of s_1 and s_2, and the algorithm will stop. As we can see from the final solution shown in Figure 3(b), the SSA algorithm selects three nodes to provide s_1 and s_2 with one node reduced.

Input: Node-service bipartite graph $G = (P, S, E)$ and requirement for service data accuracy $a_1, a_2, \ldots, a_m$.
Output: Node-service selection bipartite graph $G' = (P', S, E')$.

1: $P' \leftarrow \emptyset$, $E' \leftarrow \emptyset$, $P'_j \leftarrow \emptyset$, $j = 1, \ldots, m$;
2: **for** each service $s_j \in S$
3: **while** $\widehat{a}_j(P'_j) < a_j$
4: Calculate data accuracy increment $I_{i,j}$ for each p_i, $(p_i, s_j) \in E$ and $(p_i, s_j) \notin E'$, and select the one p_i with maximum $I_{i,j}$ and $I_{i,j} > 0$ as the candidate node; if two nodes have identical increment, the node included in P' is prior to be selected;
5: If no such a candidate is found, the algorithm stops with no solution found;
6: $P'_j \leftarrow P'_j + \{p_i\}$, $E' \leftarrow E' + (p_i, s_j)$;
7: **end while**
8: $P' \leftarrow P' \bigcup P'_j$;
9: **end for**

Algorithm 1: Pseudocode for Separate Selection Algorithm (SSA).

5.2. Combined Selection Algorithm (CSA). The previous proposed SSA tries to select nodes for each separate service based on the criterion of data accuracy increment. However, some nodes will provide several services simultaneously in the wireless sensor networks, and this multiservice property can help to improve the performance of node selection strategies if we simply select one multiservice node to improve the data accuracy required by different services. As we can see from Line 4 in Algorithm 1, we intend to select the candidate node that is already chosen to provide some other service in SSA algorithm, which means that nodes with multiservice property are more preferred in SSA algorithm during node selection process. However, this separate selection process is not always efficient especially in some cases. For example, the sample network given in Figure 3 can obtain a better solution than the solution found by SSA. If we do not select p_1 and p_4 which have maximum data accuracy increment for one special service, but select p_2 and p_3 which can provide service s_1 and s_2 simultaneously, it is obvious that this solution can guarantee the required data accuracy of s_1 and s_2. However, this solution has fewer nodes than SSA because it only selects two nodes with two nodes reduced. In this section, we will introduce a new Combined Selection Algorithm (CSA) that has utilized multiservice property for node selection problem in service-oriented wireless sensor networks.

In this paper, we aim at minimizing the number of selected nodes with the data accuracy guaranteed for all services in the network. There are two important factors that will influence the number of selected nodes; that is, the number of services and the service quality of each node. Intuitively, it helps to reduce the number of selected nodes in case that they can provide more kinds of services since more nodes are potential candidates during the selection process. However, nodes might have poor data accuracy when they are far away from the sensing source although they can provide the required services. It means that we shall consider the data accuracy as well as the number of services simultaneously during the node selection process.

The basic idea of the CSA is described as follows. Initially no nodes are selected and the final solution is an empty set. Then, nodes are chosen and added into the final solution in a sequence way. In each step, we intend to select the node with maximal *contribution increment* to all services in the network (which will be discussed in details below in this section). This process continues until the data accuracy for all services is satisfied, and finally we can obtain the selected nodes as well as the services provided by each node for the problem.

Assume that p_i is considered to provide services s_j in the current selection bipartite graph G'. In case that $I_{i,j} = 0$, it is obvious that there is no benefit for node p_i to provide s_j, and we have $(p_i, s_j) \notin E'$. In case that $I_{i,j} > 0$, we can see that node p_i helps to improve data accuracy of service s_j, and we have $(p_i, s_j) \in E'$. In this way, for a given p_i, we can calculate the $I_{i,j}$ for each s_j. Generally, we intend to choose the node that has more contribution increment to the data accuracy. Here we have introduced $w_{i,j}$ $(0 < w_{i,j} \leq 1)$ as a coefficient to demonstrate the impact of current contribution increment on the final data accuracy, and it is formulated as

$$w_{i,j} = \begin{cases} 0, & I_{i,j} = 0, \\ \exp\left\{-\left(\dfrac{\left(a_j - \widehat{a}_j\left(P'_j\right) - I_{i,j}\right)}{a_j}\right)\right\}, & I_{i,j} > 0. \end{cases} \tag{14}$$

As we can see from the above formulation, $w_{i,j} = 0$ in case that $I_{i,j} = 0$, which shows that node p_i has no contribution to the data accuracy of service s_j. Here we adopt the power exponential function to indicate how much the data accuracy is close to the requirement value.

Let C_i be the contribution increment of node p_i with the current selection bipartite graph G' in case that p_i is selected to provide services, and C_i is formulated as

$$C_i = \sum_{(p_i, s_j) \in E} w_{i,j} I_{i,j}, \quad \forall p_i \in P, \tag{15}$$

where $w_{i,j}$ $(0 < w_{i,j} \leq 1)$ is a coefficient and $I_{i,j}$ denotes the data accuracy increment for service s_j in case that node p_i

Input: Node-service bipartite graph $G = (P, S, E)$ and requirement for service data accuracy $a_1, a_2, \ldots, a_m$.
Output: Node-service selection bipartite graph $G' = (P', S, E')$.

1: $P' \leftarrow \emptyset$, $E' \leftarrow \emptyset$, $P'_j \leftarrow \emptyset$, $j = 1, \ldots, m$;
2: **while** true
3: If for each service $s_j \in S$, has satisfied $\widehat{a}_j(P'_j) \geq a_j$, the algorithm stops with solution found;
4: Calculate data accuracy increment $I_{i,j}$ and contribution increment C_i for each p_i, $p_i \in P$ and $p_i \notin P'$, and select the one p_i with maximum C_i and $C_i > 0$ as the candidate node;
5: If no such a candidate is found, the algorithm stops with no solution found;
6: **for** each service s_j, $(p_i, s_j) \in E$ and $I_{i,j} > 0$
7: $P'_j \leftarrow P'_j + \{p_i\}$, $E' \leftarrow E' + (p_i, s_j)$;
8: **while** $P'_j \neq \emptyset$
9: Calculate $\widehat{a}_j(P'_j - \{p_k\})$ for each p_k, $p_k \in P'_j$, and select the one p_k with maximum $\widehat{a}_j(P'_j - \{p_k\})$ and $\widehat{a}_j(P'_j - \{p_k\}) \geq \widehat{a}_j(P'_j)$;
10: If no such a candidate is found, **break**;
11: $P'_j \leftarrow P'_j - \{p_k\}$, $E' \leftarrow E' - (p_k, s_j)$;
12: **end while**
13: **end for**
14: $P' \leftarrow \bigcup_{j=1}^{m} P'_j$;
15: **end while**

Algorithm 2: Pseudocode for Combined Selection Algorithm (CSA).

is selected to provide service s_j, which is as same as that in Section 5.1.

So far we have introduced the basic node selection process for the CSA. However, the algorithm can be further optimized. In case that the algorithm selects one node with maximum contribution increment, this node will provide each service that helps to improve the data accuracy. However, the node with multiservice and maximum contribution increment might have poor data accuracy increment for some services. Although the node that provides poor data accuracy increment can still improve the data accuracy, the data accuracy increment is so small that it needs to select more nodes to guarantee the required data accuracy. Therefore, we can further reduce the number of nodes by removing some already selected nodes that are with poor data accuracy increment for some services. Let us consider an example that two services are supported by two nodes, in which p_1 can support s_1 and s_2, p_2 can support s_2, and the required data accuracy for s_1 and s_2 is both 0.8. Suppose that the data accuracy of s_1 provided by p_1 is 0.8, the data accuracy of s_2 provided by p_1 is 0.3, provided by p_2 is 0.8, and provided by p_1 and p_2 is 0.7. In the first selection, the value of C_1 is larger than C_2 according to formula (15), then p_1 will be selected and provide service s_1 and s_2. In the next selection, p_2 will be selected and provide service s_2. However, the data accuracy of s_2 provided by p_1 and p_2 is less than that provided by p_2. It is clear that we can improve the data accuracy of service s_2 if we let p_1 do not provide s_2. Note that the "bad" assignments (i.e., assigning nodes to provide services that are with poor data accuracy increment) cannot be eliminated during the selection process, due to the fact that they still help to improve the data accuracy. After selecting a new node, we can check all the selected nodes to find the "bad" assignments that were included in the previous selections. The basic idea of the optimization process is described as follows. In case that there is a special service for example, s_j, has selected a new node, we will firstly calculate $\widehat{a}_j(P'_j - \{p_i\})$ for each node $p_i \in P'_j$, and select the one p_i with maximum $\widehat{a}_j(P'_j - \{p_i\})$ and $\widehat{a}_j(P'_j - \{p_i\}) \geq \widehat{a}_j(P'_j)$; after that, we let p_i do not provide s_j. This process continues until no more of this kind of nodes can be found from P'_j.

The pseudocode for CSA is listed in Algorithm 2. In Line 1, the set of selected nodes P'_j for each service s_j is initialized as $\emptyset$. In Lines 2–15, nodes are selected in a sequence way until data accuracy is guaranteed for all services. In case that there is some service that current data accuracy cannot satisfy its requirement, we will firstly calculate all the candidate nodes' contribution increment, and select one node with maximum contribution increment as the candidate (in Line 4). Secondarily, the candidate node is assigned to provide each service s_j that helps to improve the data accuracy (in Line 7). Finally, we check all the nodes in P'_j and remove nodes from P'_j without declining the data accuracy of service s_j, and this subprocess continues until no more of this kind of nodes can be found from P'_j (in Line 8–12). The node selection process continues until enough nodes are selected for all services.

We illustrate the execution of CSA algorithm during one round of the iteration process by an example given in Figure 4. In this example two services are supported by four nodes. The node-service bipartite graph $G = (P, S, E)$ is given in Figure 4(a), and the current node-service selection bipartite graph $G' = (P', S, E')$ is given in Figure 4(b). As we can see from Figure 4(b), the algorithm has selected p_2 in G', then the available nodes are p_1, p_3, and p_4. Suppose that the contribution increment of each available node has been

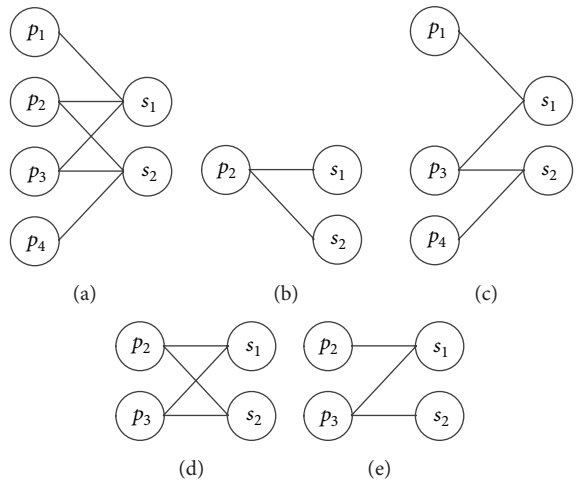


Figure 4: An example of the execution of CSA algorithm during one round of the iteration process. (a) The node-service bipartite graph $G = (P, S, E)$. (b) The current node-service selection bipartite graph $G' = (P', S, E')$. (c) The relationship between available nodes and services, and there is an edge between p_i and s_j indicating that p_i helps to improve the data accuracy of s_j. (d) Select p_3 to provide s_1 and s_2. (e) The final node-service selection bipartite graph G' of this round of the iteration process.

calculated and the value of C_1, C_3, and C_4 is 0.1, 0.3, and 0.2, respectively. The relationship between available nodes and services is given in Figure 4(c), and there is an edge between p_i and s_j indicating that the data accuracy increment of p_i is larger than 0, that is, p_i helps to improve the data accuracy of s_j. In the next step, the algorithm will select one node with maximum contribution increment, that is, p_3 in this example. According to Figure 4(c), p_3 helps to improve the data accuracy of s_1 and s_2. Then p_3 will provide s_1 and s_2, and the new solution is given in Figure 4(d). After a new node is added into the solution, the algorithm will execute an optimization process. We assume that there is one "bad" assignment (p_2, s_2); that is, the data accuracy of s_2 provided by p_3 is not less than that provided by p_2 and p_3. Then the algorithm will remove the assignment (p_2, s_2) from the solution. The final solution of this round of the iteration process can be observed from Figure 4(e).

5.3. Complexity Analysis

Lemma 1. *The time complexity of SSA is $O(mn^2)$.*

Proof. During the outside for loop, the algorithm selects nodes for each service $s_j \in S$, and each execution of the outside for loop contains a while loop. In the while loop, the algorithm checks each node p_i, $(p_i, s_j) \in E$ and $(p_i, s_j) \notin E'$ and selects one node with maximum data accuracy increment. The while loop will continue until the data accuracy of s_j is satisfied. Because there are at most n nodes that can provide s_j and each execution selects one node, the execution of the while loop takes $O(n^2)$ time. Hence, the time complexity of SSA is $O(mn^2)$. □

Lemma 2. *The time complexity of CSA is $O(mn^3)$.*

Proof. During the outside while loop, the algorithm will firstly check each node p_i, $p_i \in P$, $p_i \notin P'$, and calculate the data accuracy increment for each services and node's contribution increment. Because each node can provide at most m services and each execution selects one node with maximum contribution increment, this process takes at most $O(mn)$ time. In the next step, the loop is executed to assign the selected node to provide each service that helps to improve the data accuracy, and there are at most m services. In each execution of the inside while loop, the algorithm checks each node in P'_j and tries to find one node that without declining the data accuracy of s_j when this node does not provide s_j. The inside while loop will continue until no more of this kind of nodes can be found. Because each P'_j contains at most n nodes and each execution of the inside while loop selects one node, the inside while loop takes at most $O(n^2)$ time and for loop is $O(mn^2)$. Therefore, each execution of the outside while loop takes at most $O(mn + mn^2) = O(mn^2)$ time. Because there are at most n nodes to be selected, the time complexity of CSA is $O(mn^3)$. □

6. Simulation Results and Analysis

In order to evaluate the actual behavior of the above algorithms, we have relied on the experimental simulation to show its performance. In this section, we have firstly introduced the building process of our simulation and then analyze the impact of spatial correlation parameters θ and SNR β on the results. Finally, we compare the performance of SSA and CSA in different environments.

6.1. Simulation Setup. We use MATLAB as the platform tool that is used popularly in simulation of wireless networks. The scenarios are built in a square area 500 m × 500 m. The sensor nodes are random placed as well as the sensing sources. Here we assume that each sensing source is dedicated to one special service. Given the sensor nodes and the sensing sources, in the next step we need to decide the set of services provided by these nodes. Here we adopt the randomly model to determine whether node p_i can provide service s_j with a given probability ratio q $(0 < q < 1)$; that is, p_i only provides s_j in case that the random value (between 0 and 1) is larger than q. Here we also assume that each service is provided by at least one node. Otherwise, the scenario is rebuilt until this constraint is satisfied. And the data accuracy $\widehat{a}$ for each service is assumed to be identical. In this work, we build 100 different scenarios and compare the average performance of the proposed algorithms.

6.2. Impact of Spatial Correlation Parameters θ and SNR β. In this part, we analyze the impact of spatial correlation parameters θ and SNR β on the performance of the SSA and CSA. The spatial correlation parameters θ and SNR β are two

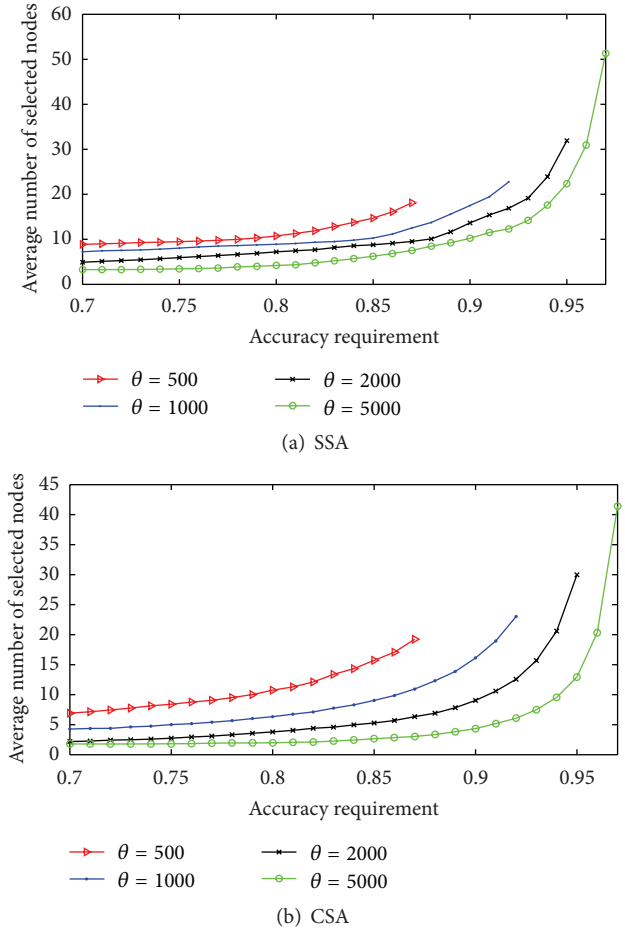


Figure 5: Impact of spatial correlation parameter θ, $\theta \in \{500, 1000, 2000, 5000\}$.

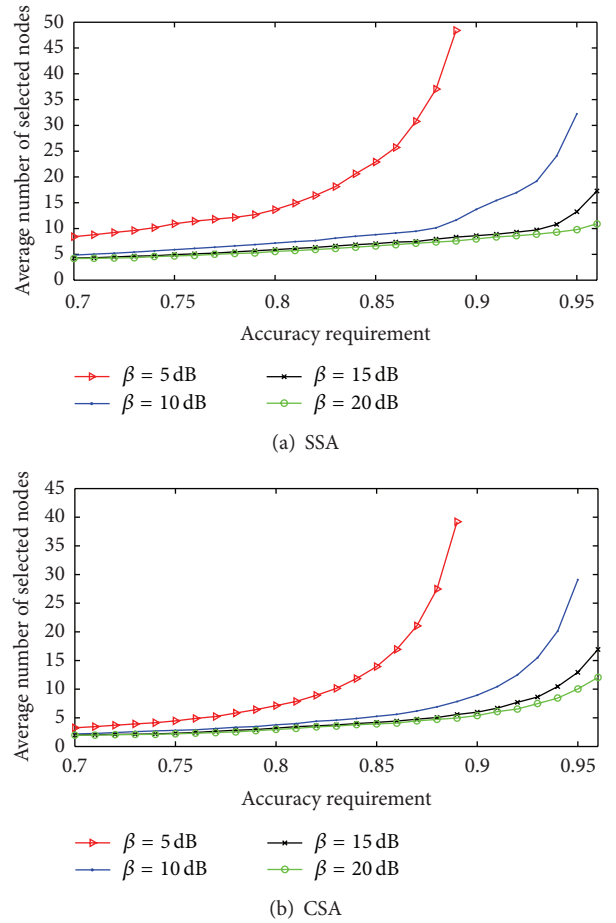


Figure 6: Impact of SNR β, $\beta \in \{5\text{ dB}, 10\text{ dB}, 15\text{ dB}, 20\text{ dB}\}$.

parameters in the spatial correlation model, which we have introduced in Section 3.2. The spatial correlation parameter θ denotes the correlation of sensed data between the distances among sensor nodes. As we can see from formula (2), the larger θ indicates a high degree of spatial correlation; that is, the nodes in a network provide strongly correlated service. The SNR β denotes the noise strength that will affect the distortion of service. It is obvious that the larger β will result in low distorted sensed data; that is, the services provided by nodes are more accurate. As we can see, the two parameters θ and β will affect the sensed data, which in turn influences the selection results.

The first set of experiments is concerned with the impact of spatial correlation parameter θ on the number of selected nodes. The simulation is done with 300 nodes and 10 services, and the SNR β is assumed to be 10 dB and q be 0.5. The spatial correlation parameter θ varies from 500, 1000, and 2000 to 5000, and we study the average number of the selected nodes compared with the change of services' accuracy requirement $\widehat{a}$ that starts from 0.7 to 0.97. As we can see from Figure 5, the number of selected nodes is minimized in case that $\widehat{a} = 0.7$, and it increases together with the increasing of accuracy requirement $\widehat{a}$. However, this process is not so significant until $\widehat{a}$ reaches some special point. For example, the average number of selected nodes is among 1.79 to 4.34 when $0.7 \leq \widehat{a} \leq 0.9$ in case that $\theta = 5000$ and using CSA algorithm to find solution; however, it increases rapidly when $\widehat{a} > 0.9$. Moreover, there might have not been enough nodes to support the required data accuracy requirement; for example, the maximum data accuracy requirement is about 0.87 in case that $\theta = 500$.

The second set of experiments is concerned with the impact of signal-to-noise ratio β on the number of selected nodes, which is illustrated in Figure 6. The simulation is done with 300 nodes and 10 services, and spatial correlation parameter θ is assumed to be 2000 and q be 0.5. The SNR parameter β varies from 5 dB, 10 dB, and 15 dB to 20 dB, and we study the average number of selected nodes compared with the change of services' accuracy requirement $\widehat{a}$ that starts from 0.7 to 0.96. We also can see that the number of selected nodes remains stable or varies linearly when $\widehat{a}$ is smaller; however, it increases rapidly when $\widehat{a}$ is larger than some special point. This conclusion is similar to that of Figure 5. As we know, the energy budget is an important criterion for the wireless sensor networks, and it will worsen the network performance if too many nodes are involved in the data sensing process. The compromise from a given application

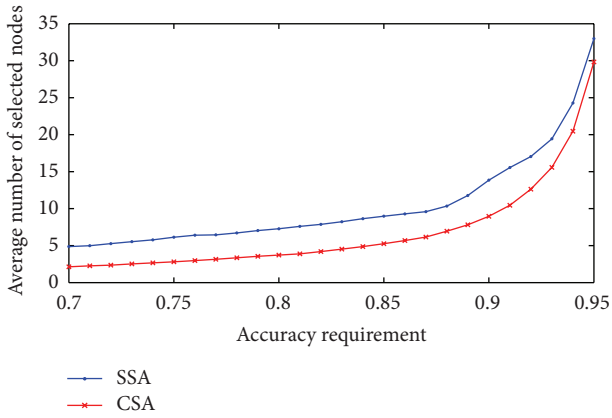


Figure 7: Comparison of SSA and CSA with different data accuracy requirements.

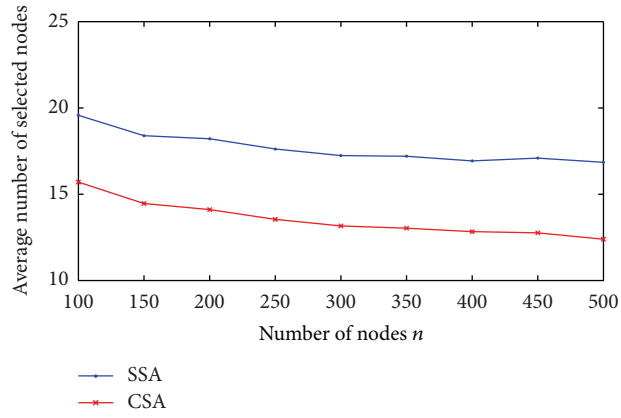


Figure 8: Comparison of SSA and CSA with different network size.

scenario will help to reduce the energy consumption by selecting a proper accuracy requirement.

6.3. Performance Comparison between SSA and CSA. So far as we know, this is the first works concerned with the node selection algorithms with data accuracy guaranteed for the service-oriented wireless sensor networks. Most of the related works [28–31] focused on different research issues, such as target tracking, and topology control. Wang et al. [13] had proposed a scheduling algorithm for the service-oriented wireless sensor network, but it did not consider the data accuracy. In this section, we compare the performance of SSA and CSA in different scenarios with varied accuracy requirement, number of nodes n, number of service m, and the value of q, respectively.

Figure 7 has shown the number of selected nodes with SSA and CSA when the accuracy requirement $\hat{a}$ varies from 0.7 to 0.95. The simulation is done with 300 nodes and 10 services, and spatial correlation parameter θ is assumed to be 2000, SNR β to be 10 dB, and q to be 0.5. The experimental results show that CSA has better performance compared with SSA in all situations.

The second set of simulations is done to show the impact of network size on the number of selected nodes. The simulation is done with 300 nodes and 10 services, and spatial correlation parameter θ is assumed to be 2000, data accuracy requirement $\hat{a}$ to be 0.92, SNR β to be 10 dB, and q be 0.5. And the network size varies from 100 to 500. As we can see from Figure 8, the CSA has better performance than SSA in all cases. Furthermore, we have two observations from Figure 8. (1) The average number of the selected nodes is relatively smaller in case that the network size is larger. This is due to the fact that there are more potential candidates for a given service with the network size increasing, and it helps to reduce the number of selected nodes. (2) The number of the selected nodes decreases slightly in case that the network size reaches some special point. It implies that it is helpless to reduce the number of selected nodes by adding more nodes into the network.

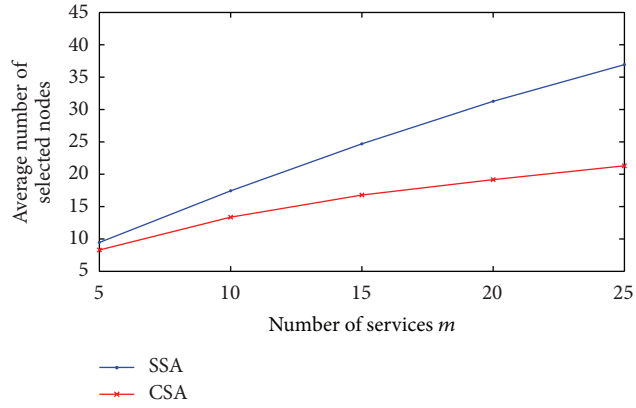


Figure 9: Comparison of SSA and CSA with different number of services.

The third set of simulations focuses on the impact of the number of services in the network on the number of selected nodes. We use the similar parameters in the second set of simulations. As we can see from Figure 9, CSA runs better than SSA with different value of m although it is not so significant when m is close to 5.

The fourth set of simulations focuses on the probability q on the number of selected nodes by varying q from 0.1 to 0.9. We also use the similar parameters in the second set of simulations. In fact, the parameter q indirectly represents the number of services provided by nodes in the network. As we can see from Figure 10, the number of the selected nodes is rather close with SSA and CSA when q is small enough. Particularly, the SSA is even slightly better than CSA when $q = 0.1$. However, the CSA shows better performance when the value of q increases. Meanwhile, we can also obtain two conclusions from this set of experiments. (1) The average number of the selected nodes decreases with q increasing. The larger q results in more services that can be provided by selected nodes. Thus, each node can make more contribution to the required services, which in turn reduces the total number of selected nodes. (2) In case that q is larger enough,

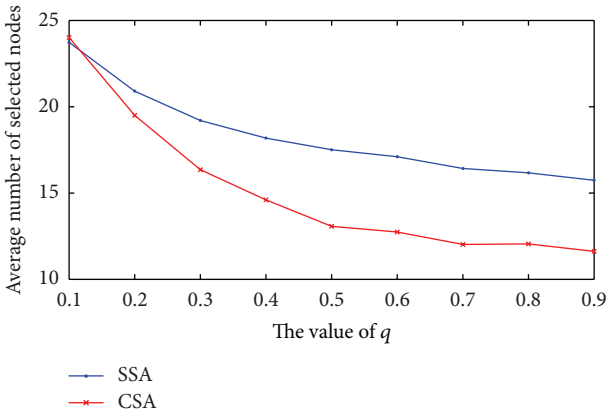


FIGURE 10: Comparison of SSA and CSA in different value of q.

for example, $q > 0.5$, the average number of selected nodes decreases slowly with q increasing.

7. Conclusion

To provide various services is one important trend for the future wireless sensor networks, and the service-oriented architecture allows different services supported simultaneously in the same physical area in which one sensor can provide different kinds of service. Quality of services, such as data accuracy, is one of the key criterions for applications because the sensed data is generally a noisy version of the physical phenomenon. The spatial correlation among the sensed data makes it possible to select a subset of nodes to provide the required services while the data accuracy is guaranteed, which is obviously helpful to improve the performance of the wireless sensor networks. We are concerned with this issue in this paper and have formulated the node selection problem into an Integer Nonlinear Programming (INLP) problem. We also have developed two heuristic algorithms, namely, Separate Selection Algorithm (SSA) and Combined Selection Algorithm (CSA) for the problem. In the future work we are to develop efficient scheduling schemes for the node selection process and aim at providing a solution for the service-oriented wireless sensor networks with the network lifetime maximized. The temporal correlation is also important to optimize the network performance. We also plan to explore energy-efficient scheduling schemes for service-oriented wireless sensor networks with both spatial and temporal correlation considered.

Acknowledgments

This work is supported by Fujian Provincial Natural Science Foundation of China under Grant no. 2011J01345, the Development Foundation of Educational Committee of Fujian Province under Grant no. 2012JA12027, the National Science Foundation of China under Grant no. 61103275, and the Technology Innovation Platform Project of Fujian Province under Grant no. 2009J1007.

References

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks*, vol. 38, pp. 393–422, 2002.

[2] D. Gračanin, M. Eltoweissy, A. Wadaa, and L. A. DaSilva, "A service-centric model for wireless sensor networks," *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 6, pp. 1159–1166, 2005.

[3] X. Chu and R. Buyya, "Service oriented sensor web," in *Sensor Network and Configuration: Fundamentals, Techniques, Platforms, and Experiments*, N. P. Mahalik, Ed., pp. 51–74, 2007.

[4] A. Rezgui and M. Eltoweissy, "Service-oriented sensor-actuator networks: promises, challenges, and the road ahead," *Computer Communications*, vol. 30, no. 13, pp. 2627–2648, 2007.

[5] J. King, R. Bose, I. Y. Hen, S. Pickles, and A. Helal, "Atlas: a service-oriented sensor platform hardware and middleware to enable programmable pervasive spaces," in *Proceedings of the 31st Annual IEEE Conference on Local Computer Networks (LCN '06)*, pp. 630–638, Tampa, Fla, USA, November 2006.

[6] Crossbow Technology, http://www.xbow.com/.

[7] J. Li and S. Cheng, "(ε, σ)-approximate aggregation algorithms in dynamic sensor networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, pp. 385–396, 2012.

[8] C. Wang, H. Ma, Y. He, and S. Xiong, "Adaptive approximate data collection for wireless sensor networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, pp. 1004–1016, 2012.

[9] M. C. Vuran, Ö. B. Akan, and I. F. Akyildiz, "Spatio-temporal correlation: theory and applications for wireless sensor networks," *Computer Networks*, vol. 45, no. 3, pp. 245–259, 2004.

[10] X. Dong and M. C. Vuran, "Spatio-temporal soil moisture measurement with wireless underground sensor networks," in *Proceedings of the IFIP Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net '10)*, pp. 1–8, Juan Les Pins, France, June 2010.

[11] E. Avilés-López and J. García-Macías, "TinySOA: a service-oriented architecture for wireless sensor networks," *Service Oriented Computing and Applications*, vol. 3, pp. 99–108, 2009.

[12] J. M. Corchado, J. Bajo, D. I. Tapia, and A. Abraham, "Using heterogeneous wireless sensor networks in a telemonitoring system for healthcare," *IEEE Transactions on Information Technology in Biomedicine*, vol. 14, no. 2, pp. 234–240, 2010.

[13] J. Wang, D. Li, G. Xing, and H. Du, "Cross-layer sleep scheduling design in service-oriented wireless sensor networks," *IEEE Transactions on Mobile Computing*, vol. 9, no. 11, pp. 1622–1633, 2010.

[14] X. Wang, J. Wang, Z. Zheng, Y. Xu, and M. Yang, "Service composition in service-oriented wireless sensor networks with persistent queries," in *Proceedings of the 6th IEEE Consumer Communications and Networking Conference (CCNC '09)*, pp. 1–5, Las Vegas, Nev, USA, January 2009.

[15] P. Wang and I. F. Akyildiz, "Spatial correlation and mobility-aware traffic modeling for wireless sensor networks," *IEEE/ACM Transactions on Networking*, vol. 19, pp. 1860–1873, 2011.

[16] Y. Ma, Y. Guo, X. Tian, and M. Ghanem, "Distributed clustering-based aggregation algorithm for spatial correlated sensor networks," *IEEE Sensors Journal*, vol. 11, no. 3, pp. 641–648, 2011.

[17] S. Pattem, B. Krishnamachari, and R. Govindan, "The impact of spatial correlation on routing with compression in wireless sensor networks," *ACM Transactions on Sensor Networks*, vol. 4, no. 4, article 24, 2008.

[18] J. Faruque and A. Helmy, "RUGGED: routing on fingerprint gradients in sensor networks," in *Proceedings of IEEE International Conference on Pervasive Services (ICPS '04)*, pp. 179–188, Beirut, Lebanon, July 2004.

[19] M. C. Vuran and Ö. B. Akan, "Spatio-temporal characteristics of point and field sources in wireless sensor networks," in *Proceedings of IEEE International Conference on Communications (ICC '06)*, pp. 234–239, Istanbul, Turkey, July 2006.

[20] D. Zordan, G. Quer, M. Zorzi, and M. Rossi, "Modeling and generation of space-time correlated signals for sensor network fields," in *Proceedings of IEEE Global Telecommunications Conference (GLOBECOM '11)*, pp. 1–6, Houston, Tex, USA, December 2011.

[21] A. Jindal and K. Psounis, "Modeling spatially correlated data in sensor networks," *ACM Transactions on Sensor Networks*, vol. 2, no. 4, pp. 466–499, 2006.

[22] X. Han, X. Cao, E. L. Lloyd, and C. C. Shen, "Fault-tolerant relay node placement in heterogeneous wireless sensor networks," *IEEE Transactions on Mobile Computing*, vol. 9, no. 5, pp. 643–656, 2010.

[23] T. Banerjee, B. Xie, and D. P. Agrawal, "Fault tolerant multiple event detection in a wireless sensor network," *Journal of Parallel and Distributed Computing*, vol. 68, no. 9, pp. 1222–1234, 2008.

[24] N. Xiong, A. Vasilakos, L. Yang et al., "Comparative analysis of quality of service and memory usage for adaptive failure detectors in healthcare systems," *IEEE Journal on Selected Areas in Communications*, vol. 27, no. 4, pp. 495–509, 2009.

[25] N. Xiong, A. V. Vasilakos, J. Wu et al., "A self-tuning failure detection scheme for cloud computing service," in *Proceedings of IEEE 26th International Parallel & Distributed Processing Symposium (IPDPS '12)*, pp. 668–679, Shanghai, China, May 2012.

[26] G. Wu, C. Lin, F. Xia, L. Yao, H. Zhang, and B. Liu, "Dynamical jumping real-time fault-tolerant routing protocol for wireless sensor networks," *Sensors*, vol. 10, no. 3, pp. 2416–2437, 2010.

[27] J. Bu, M. Yin, D. He, F. Xia, and C. Chen, "SEF: a secure, efficient, and flexible range query scheme in two-tiered sensor networks," *International Journal of Distributed Sensor Networks*, vol. 2011, Article ID 126407, 12 pages, 2011.

[28] Y. Cai, W. Lou, M. Li, and X. Y. Li, "Energy efficient target-oriented scheduling in directional sensor networks," *IEEE Transactions on Computers*, vol. 58, no. 9, pp. 1259–1274, 2009.

[29] J. Lin, W. Xiao, F. L. Lewis, and L. Xie, "Energy-efficient distributed adaptive multisensor scheduling for target tracking in wireless sensor networks," *IEEE Transactions on Instrumentation and Measurement*, vol. 58, no. 6, pp. 1886–1896, 2009.

[30] M. P. Johnson, H. Rowaihy, D. Pizzocaro et al., "Sensor-mission assignment in constrained environments," *IEEE Transactions on Parallel and Distributed Systems*, vol. 21, no. 11, pp. 1692–1705, 2010.

[31] Y. Liu, L. Ni, and C. Hu, "A generalized probabilistic topology control for wireless sensor networks," *IEEE Journal on Selected Areas in Communications*, vol. 30, no. 9, pp. 1780–1788, 2012.

[32] E. Fasolo, M. Rossi, J. Widmer, and M. Zorzi, "In-network aggregation techniques for wireless sensor networks: a survey," *IEEE Wireless Communications*, vol. 14, no. 2, pp. 70–87, 2007.

[33] W. Zhu and H. Fan, "A discrete dynamic convexized method for nonlinear integer programming," *Journal of Computational and Applied Mathematics*, vol. 223, no. 1, pp. 356–373, 2009.

[34] W. Zhu and G. Lin, "A dynamic convexized method for nonconvex mixed integer nonlinear programming," *Computers and Operations Research*, vol. 38, no. 12, pp. 1792–1804, 2011.

[35] M. Schlüter, J. A. Egea, and J. R. Banga, "Extended ant colony optimization for non-convex mixed integer nonlinear programming," *Computers and Operations Research*, vol. 36, no. 7, pp. 2217–2229, 2009.

Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2013, Article ID 985410, 14 pages
http://dx.doi.org/10.1155/2013/985410

Research Article

A PSO-Optimized Minimum Spanning Tree-Based Topology Control Scheme for Wireless Sensor Networks

Wenzhong Guo,[1,2] Bin Zhang,[1] Guolong Chen,[1] Xiaofeng Wang,[2] and Naixue Xiong[3]

[1] *College of Mathematics and Computer Science, Fuzhou University, Fuzhou 350108, China*
[2] *College of Computer, National University of Defense Technology, Changsha 410073, China*
[3] *School of Computer Science, Colorado Technical University, Colorado Spring, CO 80907, USA*

Correspondence should be addressed to Guolong Chen; fzucgl@163.com

Received 6 January 2013; Accepted 15 March 2013

Academic Editor: Hongju Cheng

Wireless sensor networks (WSNs) are networks of autonomous nodes used for monitoring an environment. Topology control is one of the most fundamental problems in WSNs. To overcome high connectivity redundancy and low structure robustness in traditional methods, a PSO-optimized minimum spanning tree-based topology control scheme is proposed in this paper. In the proposed scheme, we transform the problem into a model of multicriteria degree constrained minimum spanning tree (mcd-MST) and design a nondominated discrete particle swarm optimization (NDPSO) to deal with this problem. To obtain a better approximation of true Pareto front, the multiobjective strategy with a fitness function based on niche and phenotype sharing function is applied in NDPSO. Furthermore, a topology control scheme based on NDPSO is proposed. Simulation results show that NDPSO can converge to the non-dominated front quite evenly, and the topology derived under the proposed topology control scheme has lower total power consumption, higher robust structure, and lower contention among nodes.

1. Introduction

A wireless sensor network (WSN) is a system of spatially distributed sensor nodes to collect important information in the target environment. WSNs have been envisioned for a wide range of applications, such as battlefield intelligence, environmental tracking, and emergency response. Each sensor node has limited computational capacity, battery supply, and communication capability [1]. Topology control and management—how to determine the transmission power of each node so as to maintain network connectivity while consuming the minimum possible power—are one of the most important issues in WSNs [2, 3]. Without proper topology control algorithms in placing a randomly connected multihop wireless sensor network may suffer from poor network utilization, high end-to-end delays, and short network lifetime. Topology control in WSNs is NP-hard [4], therefore approximate methods can be used to tackle it efficiently.

Generally, the topology control technologies can be divided into three types. One is to reduce the node redundancy in a given network that usually periodically let selected nodes enter energy-saving mode. Chen et al. [5] proposed a distributed algorithm in which each node makes its local decision on whether to sleep or not. Ding et al. [6] presented an adaptive partition scheme for node scheduling and topology control with the aim of reducing energy consumption. The second type of topology control is to use the clustering strategy such as the well-known LEACH [7]. It used rotation of the cluster head in order to evenly distribute the energy consumption. Cluster heads collected and aggregated all signals and then transmitted the fused information to the base station. Lindsay and Raghavendra [8] proposed PEGASIS which formed a chain including all nodes in the network. In PEGASIS, each node communicated only with a close neighbor and took turns transmitting to the base station. It was better than LEACH because it performed data aggregation at each chain node. The third type of topology control focuses on reducing link redundancy, and several topology control algorithms have been proposed. For example, Li et al. [9] introduced a cone-based topology control algorithm called CBTC which required only the availability of directional information. In this algorithm, each node tried

to find the minimum power p to ensure that the transmitting with p could reach some node in every cone of degree α. The algorithm had been analytically shown to be able to preserve the network connectivity if $\alpha < 5\pi/6$. Rodoplu and Meng [10] proposed a relay-region and enclosure-based approach (R and M) that introduced the notion of relay region and enclosures for the purpose of power control. It could guarantee connectivity of the entire network, and the resulted topology was a minimum power topology. However, the network topology given by this algorithm had a high degree of link redundancy, which would reduce the efficiency of the upper network protocols. To solve this problem, Li and Halpern [11] proposed an improved protocol called SMECN, which had lower link maintenance costs than R and M and could achieve a significant saving in energy consumption. Based on the local minimum spanning tree (MST), Li et al. [12] proposed a network topology control algorithm called LMST which could effectively reduce the power consumption. The topology derived under this algorithm could also preserve the network connectivity, and the degree of each node in the resulted topology was bounded by 6. However, this algorithm ignored the structure robustness and communication interference, so some network performance under LMST may be weaken.

In a word, all the algorithms mentioned previously have taken different approaches to study the topology control and management in WSNs. However, the structure robustness and the radio contention are often ignored. It is a challenge to design a novel topology control scheme which can overcome the defects (i.e., high redundancy of connectivity and low robustness of structure) of traditional methods. This challenge motivates us to integrate the MST method and particle swarm optimization (PSO) by developing a PSO-optimized minimum spanning tree-based topology control scheme in wireless sensor networks to prolong the network lifetime. In our previous work [13], we have transformed the problem of topology control into a model of multicriteria degree constrained minimum spanning tree (mcd-MST) and designed a MST-based topology control scheme with NDPSO called MCMST. Compared with [13], the major contributions of this study are summarized as follows.

(1) Similar to [13], the topology control problem is also transformed into the model of mcd-MST with low power consumption, high structure robustness, and low node degree and the constraint of the node degree corresponds with the low ratio interference.

(2) We do performance analysis of multiobjective PSO (MOPSO) in detailed and use three standard test functions (ZDT1, ZDT2, and ZDT6) to compare MOPSO with another two classical multiobjective evolutionary algorithms, NSGA and SPEA.

(3) In this paper, we adopt NDPSO, which is a nondominated discrete particle swarm optimization, to solve this mcd-MST problem. It aims at obtaining a better approximation of true Pareto front by applying the multiobjective strategy with a fitness function based on niche and phenotype sharing function. Moreover, we further do some extended experiments to analyze the performance of NDPSO by varying the number of objectives, number of vertexes, the correlation, and maximum permissible degree constrained.

(4) We introduce the MCMST topology control scheme in WSNs [13] and compare it with another two topology control schemes, SMECN and LMST, in different performance factors which include average link length, average radius, average link strong, average physical degree, and average logic degree.

The remainder of this paper is organized as follows. Section 2 describes the problem. In Section 3, we present the details of the proposed NDPSO algorithm for the mcd-MST problem. Section 4 introduces the proposed MCMST scheme based on NDPSO. In Section 5, we analyze the proposed NDPSO algorithm under a variety of scenarios. And the comparison of our proposed MCMST scheme and other existing schemes is carried out in the simulation. Finally, we make conclusions and discuss the future work in Section 6.

2. Problem Description

2.1. mc-MST. MST problem is to find a least-cost spanning tree in an edge-weighted graph, and multicriteria minimum spanning tree (mc-MST) problem is one of the extended formulations of the MST problem. In mc-MST, a vector of weights is defined for each edge, and the problem is to find all Pareto optimal spanning trees. Consider a connected and undirected graph $G = (V, E)$, where $V = \{v_1, v_2, \ldots, v_n\}$ is a finite set of vertices representing terminals or telecommunication stations, and so forth, and $E = \{e_{1,2}, e_{1,3}, \ldots, e_{i,j}, \ldots e_{n-1,n}\}$ which is defined as follows:

$$e_{i,j} = \begin{cases} 1, & \text{if } v_i, v_j \text{ have edge} \\ 0, & \text{otherwise,} \end{cases} \quad (i = 1, 2, \ldots, n-1;\ j = i+1, i+2, \ldots, n). \tag{1}$$

Each edge has m positive real numbers, representing m attributes defined on it and denoted with $w_{i,j} = \{w_{i,j}^1, w_{i,j}^2, \ldots, w_{i,j}^m\}$. In practice, $w_{i,j}^k$ ($k = 1, 2, \ldots, m$) may represent the distance, cost, and so on.

Let $x = x_{1,2}, x_{1,3}, \ldots, x_{i,j}, \ldots, x_{n-1,n}$ be defined as follows:

$$x_{i,j} = \begin{cases} 1, & \text{if } e_{i,j} = 1 \text{ and is selected} \\ 0, & \text{otherwise,} \end{cases} \quad (i = 1, 2, \ldots, n-1;\ j = i+1, i+2, \ldots, n). \tag{2}$$

Then a spanning tree of graph G can be expressed as the vector x. Let X be the set of all such vectors corresponding to spanning trees in graph G, and the mc-MST problem can be formulated as follows:

$$\begin{aligned} &\min \quad f_1(x) = \sum w_{i,j}^1 x_{i,j}, \\ &\min \quad f_2(x) = \sum w_{i,j}^2 x_{i,j}, \\ &\cdots \\ &\min \quad f_m(x) = \sum w_{i,j}^m x_{i,j}, \end{aligned}$$

$$\begin{aligned} &(i = 1, 2, \ldots, n-1;\ \ j = i+1, i+2, \ldots, n), \\ &\text{s.t.} \quad x \in X, \end{aligned} \tag{3}$$

where $f_i(x)$ is the ith objective to be minimized for the problem

$$1 \le \sum x_{i,j} \le d; \quad (i = 1, 2, \ldots, n), \tag{4}$$

where d denotes the maximum permissible degree.

If the degree constraint condition of formula (4) is further considered for the formula (3), the mc-MST problem will be transformed into a mcd-MST problem, thus the mcd-MST is a typical mc-MST.

Compared with the traditional MST problem, the mcd-MST problem has more than one objective. Because these multiple objectives exist and usually conflict with each other, we cannot determine which edge has the least weight and span one to form a spanning tree like the process of vertex growth or edge growth. If we transform the multiple objectives into a single objective according to some criteria, the problem can be easily solved by using any MST algorithm, but only one single solution can be obtained. Actually, this transformation is not an easy work both for the decision maker and the system analyzer in practice.

2.2. Model Formulation. A wireless sensor network can be described as a directed graph $G(V, E)$ in a two-dimensional plane, where V is the set of nodes and E is the set of edges. The relative neighbor graph $G_i(V_i, E_i)$ of node i is denoted as the subgraph located in the disk centered at node i with a radius $r_{\max}$, where $|V| = n$ and $|E| = m$. In this paper, a minimum spanning tree with low power consumption, high structure robustness, and low node degree in $G_i(V_i, E_i)$ is constructed. The constraint of node degree corresponds with the low ratio interference. The low power consumption and high structure robustness can be described as follows.

Objective 1 (low power consumption). One has

$$\text{Min}\left\{y_1(i) = \sum p_k \mid i \in V, k \in V_i\right\}, \tag{5}$$

where $y_1(i) \in (0, (n_i - 1)p_{\max})$ and p_k stands for the power consumption of node k.

Objective 2 (high structure robustness). One has

$$\text{Max}\left\{y_2(i) = \sum R_{jk} \mid (j,k) \in T_i, T_i \subset G_i(V_i, E_i)\right\}, \tag{6}$$

where R_{jk} stands for the robustness value between node j and node k.

Therefore, the mcd-MST problem with low power consumption and high structure robustness can be described as searching for sensor nodes with abundant surplus energy to communicate with each other, and avoiding too much interference among them. We denote an m-dimensional vector to describe the states of links. If an edge belongs to T_i, then $x_j = 1$, otherwise $x_j = 0$. Additionally, the original model must be consistent with the features of MST, so the mathematical model can be described as follows:

$$\begin{aligned} \min z_1(i) &= W_1 X = \sum_{j=1}^{m} w_{1j} x_j, \\ \min z_2(i) &= W_2 X = \sum_{j=1}^{m} w_{2j} x_j, \end{aligned} \tag{7}$$

where W_1, W_2 correspond to the weight of the vectors implying low power consumption and high structure robustness, respectively. At the same time, the objectives must satisfy the node degree constraint in order to form a degreeconstrained tree structure.

3. Proposed Algorithm

The MST problem has been well studied, and many efficient polynomial-time algorithms [14] have been developed by Dijkstra, Kruskal, Prim. Although the basic MST problem is in polynomial time, the addition of one or more constraints often transforms it into a multiobjective problem (MOP), and so approximate methods must be used if one is to tackle it efficiently. In [15], Zhou and Gen described a Prufer-encoded genetic algorithm (GA), which used Srinivas and Deb's Nondominated Sorting method and a Prufer based encoding [16, 17]. And an algorithm for enumerating all Pareto optimal spanning trees was put forward to evaluate their proposed GA. However, Knowles [18] pointed out that the proposed enumeration algorithm was not correct. In our previous work [19], we put forward a nongenerational multiobjective GA (MOGA) to deal with the mc-MST and presented an improved enumeration algorithm to evaluate our proposed MOGA. In [19], the improved enumeration algorithm was proved to be able to find out all true Pareto optimal solutions, so it can be used to replace the algorithm of Zhou and Gen for evaluating the quality of mc-MST solutions generated by an approximate method such as the GA in [15] or [19].

Definition 1 (dominance [20]). A vector $U = (u_1, u_2, \ldots, u_n)$ is said to dominate $V = (v_1, v_2, \ldots, v_n)$ if and only if U is partially less than V, that is, for all $i \in \{1, 2, \ldots, n\}, u_i \le v_i \wedge \exists\, i \in \{1, 2, \ldots, n\}, u_i \prec v_i$.

Definition 2 (pareto optimal [20]). A solution $X_u \in U$ is said to be Pareto optimal if and only if there is no $x_v \in V$ for which $V = f(X_v) = (v_1, v_2, \ldots, v_n)$ dominates $U = f(X_u) = (u_1, u_2, \ldots, u_n)$.

PSO is a relatively recent heuristic optimization technique developed by Kennedy and Eberhart [21]. Ease of implementation, high quality of solutions, computational efficiency, and speed of convergence are strengths of the PSO. In the past several years, PSO has been successfully applied in many researches and applications such as [22–27]. WSN issues such as node deployment, localization, energy-aware clustering, data aggregation, and topology control are often formulated as optimization problems, and PSO has been applied to

address these WSN issues. Kulkarni and Venayagamoorthy [28] introduce PSO and discuss its suitability for WSN applications. They also present a brief survey of how PSO is tailored to address these issues.

3.1. Basic PSO. PSO is a population based search problem where individuals, referred to as particles, are grouped into a swarm. In PSO, each particle is defined as a potential solution to a problem in a D-dimensional space, with the ith particle represented as $X_i = (X_{i1}, X_{i2}, \ldots, X_{iD})$. The particle adjusts its position in search space according to its own experience and that of neighboring particles. Each particle also maintains a memory (pbest) of its previous best position represented as $p_i = (p_{i1}, p_{i2}, \ldots, p_{iD})$ and a velocity along each dimension represented as $V_i = (V_{i1}, V_{i2}, \ldots, V_{iD})$. In each generation, the *pbest* vector of the particle with the best fitness in the local neighborhood designated p_{gd}.

In each generation of the early versions of PSO, the particles were manipulated according to the following equation:

$$v_{id} = w * v_{id} + c_1 r_1 \left(p_{id} - x_{id}\right) + c_2 r_2 \left(p_{gd} - x_{id}\right), \quad x_{id} = x_{id} + v_{id}, \tag{8}$$

where d is the number of dimensions (variables), w is the inertia weight, c_1 and c_2 are two positive constants, called acceleration constants, and r_1 and r_2 are two random numbers within the range $[0, 1]$. A constant $V_{\max}$ is often used to limit the velocities of the particles and improve the resolutions of the search space.

3.2. NDPSO. As (8) mentioned in the previous section, it is obvious that the basic PSO cannot be directly used to generate a discrete combinatorial solution of the MST problem. Since the PSO algorithm was proposed by Kennedy and Eberhart in 1995, attempts have been made to apply the PSO algorithm to discrete combinatorial problems lately [29–35]. In this section, inspired by the [32] and our previous work [29], NDPSO is applied to deal with mc-MST problem. In the proposed algorithm, the phenotype sharing function of the objective space is applied in the definition of fitness function to obtain a better approximation of true Pareto front.

3.2.1. Representation of Particles. Here we also adopt the method of [15, 19] to build the representation scheme of particles.

Procedure: Encoding

Step 1. Let vertex j be the smallest labeled leaf vertex in a labeled tree T.

Step 2. Set k to be the first digit in the permutation if vertex k is incident to vertex j.

Step 3. Remove vertex j and the edge from j to k; we have a tree with $n - 1$ vertices.

Step 4. Repeat the previously steps until one edge is left, and produce the Prufer number or permutation with $n - 2$ digits in order.

Procedure: Decoding

Step 1. Let P be the original Prufer number, and let P be the set of all vertices not included in P.

Step 2. Let j be the vertex with the smallest label in P, and let k be the leftmost digit of P. Add the edge from j to k into the tree. Remove j from P and k from P. If k does not occur anywhere in the remainder of P, put it into P.

Step 3. Repeat the process until no digits left in P.

Step 4. If no digits remain in P, there are exactly two vertices, r and s, in P. Add edge from r to s into the tree and form a tree with n-1edges.

3.2.2. Discrete Procedure of PSO. Here, the position of the ith particle at iteration t can be updated as follows:

$$X_i^t = c_2 \oplus F_3\left(c_1 \oplus F_2\left(w \oplus F_1\left(X_i^{t-1}\right), P_i^{t-1}\right), G_i^{t-1}\right). \tag{9}$$

The update equation consists of three components as follows, where r_1, r_2, and r_3 are uniform random numbers generated between 0 and 1.

(a) λ_i^t represents the velocity of the particle,

$$\lambda_i^t = w \oplus F_1\left(X_i^{t-1}\right) = \begin{cases} F_1\left(X_i^{t-1}\right), & r_1 < w, \\ X_i^{t-1}, & \text{else}, \end{cases} \tag{10}$$

where F_1 indicates the insert (mutation) operator with the probability of w.

(b) δ_i^t is the "cognition" part of the particle for the private thinking of the particle itself,

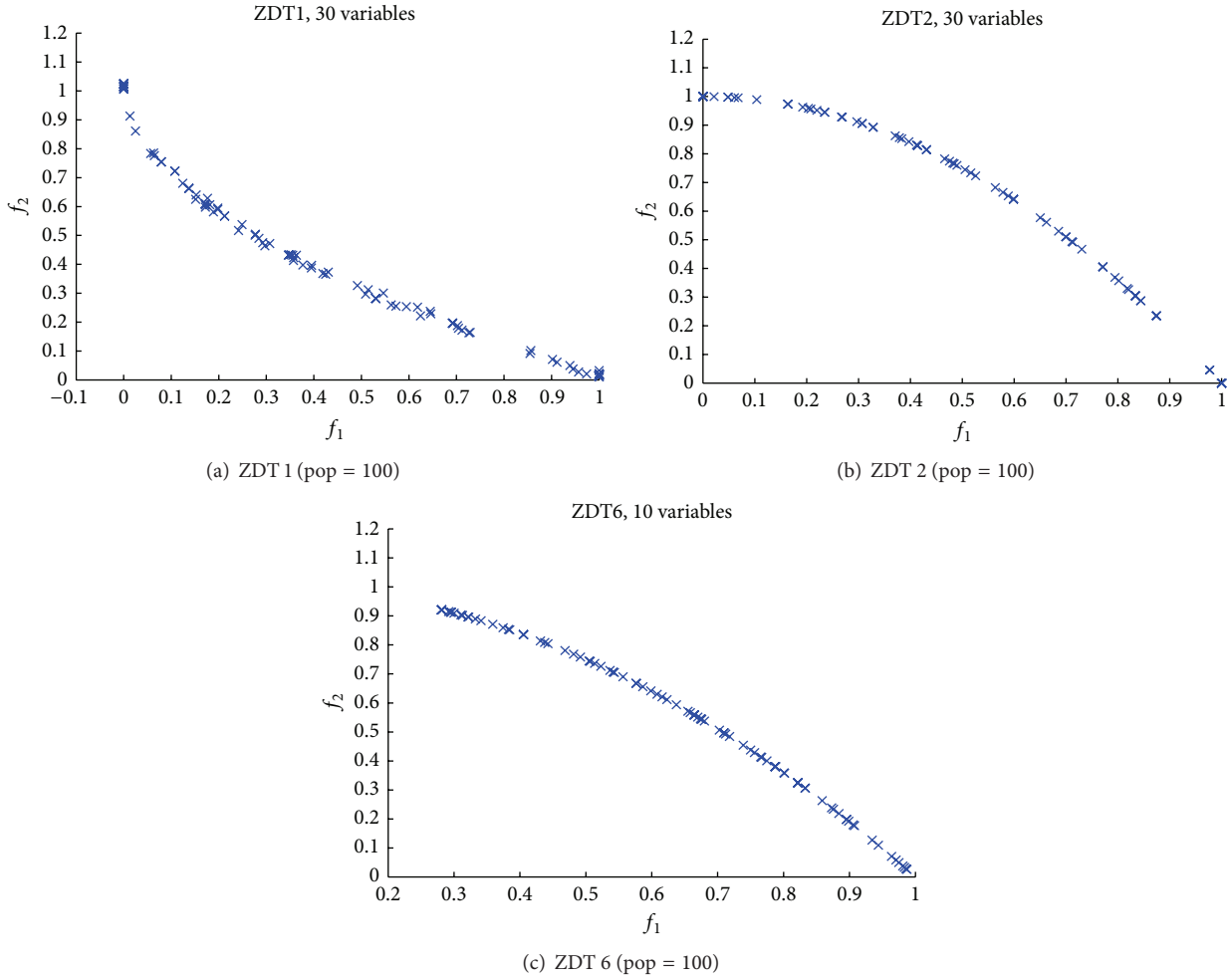
$$\delta_i^t = c_1 \oplus F_2\left(\lambda_i^t, P_i^{t-1}\right) = \begin{cases} F_2\left(\lambda_i^t, P_i^{t-1}\right), & r_2 < c_1, \\ \lambda_i^t, & \text{else}, \end{cases} \tag{11}$$

where F_2 represents the crossover operator with the probability of c_1.

(c) X_i^t is the "social" part of the particle representing the collaboration among particles,

$$X_i^t = c_2 \oplus F_3\left(\delta_i^t, G_i^{t-1}\right) = \begin{cases} F_3\left(\delta_i^t, G_i^{t-1}\right), & r_3 < c_2, \\ \delta_i^t, & \text{else}, \end{cases} \tag{12}$$

where F_3 represents the crossover operator with the probability of c_2.



(a) ZDT 1 (pop = 100)

(b) ZDT 2 (pop = 100)

(c) ZDT 6 (pop = 100)

Figure 1: The results of MOPSO in ZDT1, ZDT2, and ZDT6.

3.2.3. Fitness Value Function

Definition 3. Target distance fd_{ij}: fd_{ij} is the distance between the two particles i and j. Supposed that the distance has m dimensions which are noted as $f_1 d_{ij}, f_2 d_{ij}, \ldots, f_m d_{ij}$, respectively, and

$$fd_{ij} = f_1 d_{ij} + f_2 d_{ij} + \cdots + f_m d_{ij} = \left| f_1\left(x^i\right) - f_1\left(x^j\right) \right| + \left| f_2\left(x^i\right) - f_2\left(x^j\right) \right| + \cdots + \left| f_m\left(x^i\right) - f_m\left(x^j\right) \right|, \tag{13}$$

where $i \neq j$.

Definition 4. Dominance measure $D(i)$: $D(i)$ expresses the state of domination the ith particle with respect to the current population, and

$$D(i) = \sum_{j=1}^{p} nd(i, j), \tag{14}$$

where $nd(i, j)$ is one if particle j dominate particle i, and zero otherwise.

Definition 5. Sharing function $sh(fd_{ij})$:

$$sh\left(fd_{ij}\right) = \begin{cases} 1, & \text{if } fd_{ij} \leq \sigma_s, \\ 0, & \text{otherwise,} \end{cases} \tag{15}$$

where σ_s is a sharing parameter.

Definition 6. The neighbor density measure $N(i)$: $N(i)$ associated with particle i is defined as

$$N(i) = \sum_{j=1}^{p} sh\left(fd_{ij}\right). \tag{16}$$

Definition 7. The fitness of A given particle $F(i)$: $F(i)$ is then defined as

$$F(i) = (1 + D(i)) \times (1 + N(i)). \tag{17}$$

Compared with the single-object PSO, during the search process of multiobjective PSO (MOPSO), particles often have more than one personal best and global best value. We save these information into an external archive [36]. A proper mechanism of choosing leader particles can help to find more

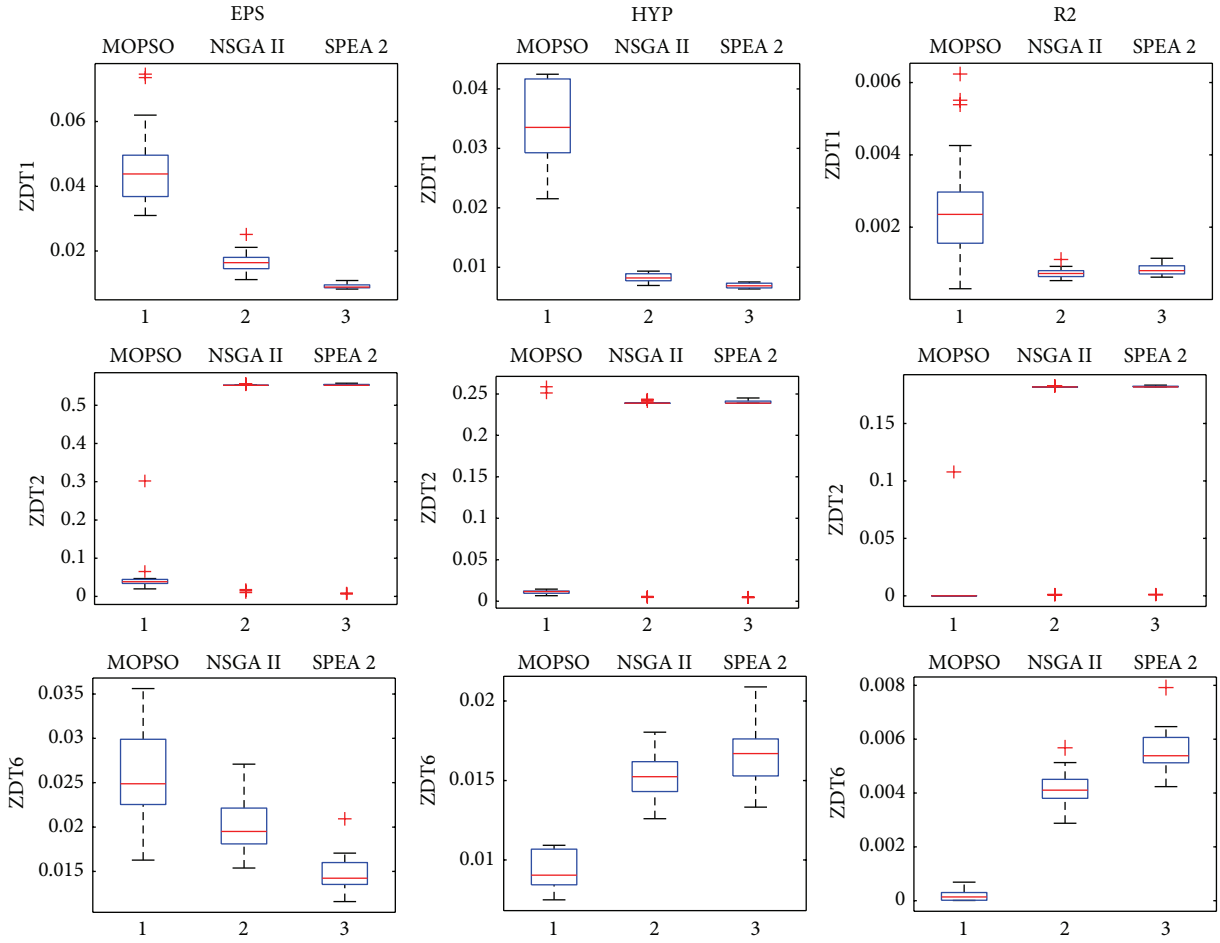
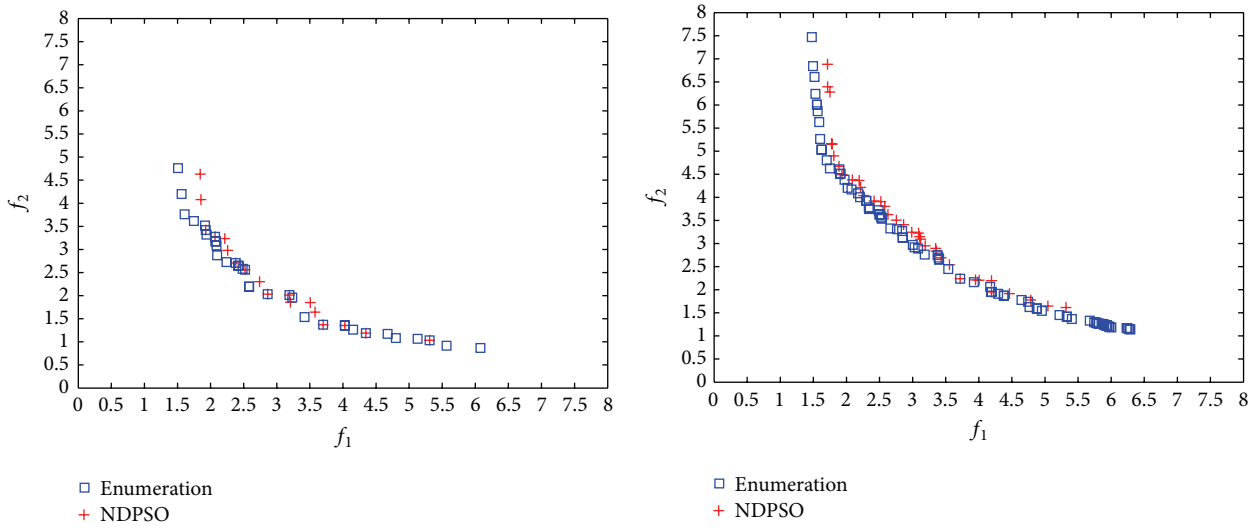


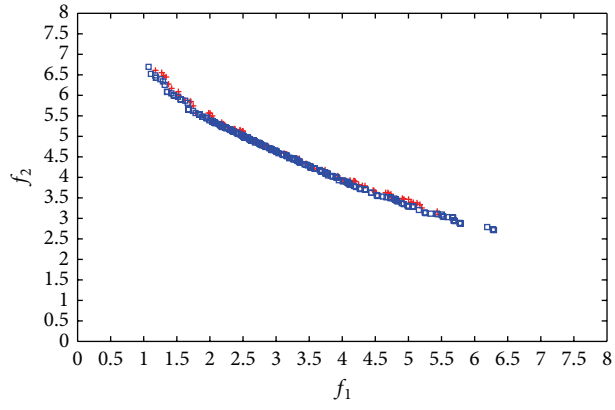
Figure 2: Boxplots of MOPSO, NSGA II, and SPEA 2 algorithm performance indicator values.



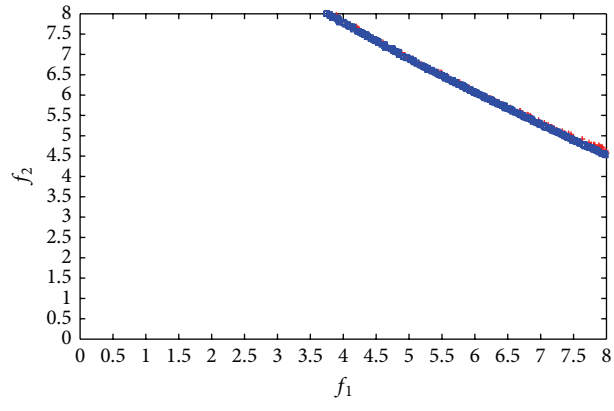
(a) mc-MST with 10 vertexes and 2 objectives

(b) mc-MST with 15 vertexes and 2 objectives

Figure 3: The effect of NDPSO with correlation = 0 on the different problem.



(a) mc-MST with 10 vertexes and 2 objectives



(b) mc-MST with 15 vertexes and 2 objectives

FIGURE 4: The effect of NDPSO with correlation = −0.7 on the different problem.

Pareto solutions in a shorter time. So it is very important to decide how to choose the proper leader particles to direct the movement of particles. In order to avoid the external archive growing too big, we adopt ε-dominance [37] to reduce the external archive.

3.2.4. Algorithm Overview. The details of NDPSO algorithm can be described as follows.

NDPSO Algorithm for mcd-MST

Step 1. Initialize swarm.

Step 2. Modify degree if not satisfied.

Step 3. Calculate the fitness value.

Step 4. Initialize leaders in an external archive.

Step 5. Filter (leaders).

Step 6. Iteration = 0.

Step 7. For each particle: mutation; select leader; SelfCross; SocialCross; modify degree; update Position; update pbest.

Step 8. Update leaders in the external archive.

Step 9. Filter (leaders).

Step 10. Iteration++.

Step 11. If iteration<iterationMax, go to Step 7.

Step 12. Output results.

4. Topology Control Scheme

As mentioned in the previous sections, we constructed the multiobjective minimum spanning tree model according to characteristics of problem and designed a NDPSO algorithm to deal with the model. Sensor nodes are usually presented in the intersection of many node adjacency coverage graphs, because the local topology is needed to adjust overall after deployed. Then the node selects the one whose requirement for coverage is maximum from a number of the spanning trees. Inspired by the idea of local minimum spanning tree structure and topology adjustment strategy of [12], a topology control scheme based on NDPSO called MCMST is proposed in this section, and the details of the MCMST are described as follows.

Topology Control Scheme Based on (MCMST)

Step 1. Each wireless sensor node u collects the local information of its own reachable neighborhood graph $G_u(V_u, E_u)$ through broadcasting a HELLO message using its maximal transmission power periodically.

Step 2. According to the information collected in the previously step and the performance evaluation function, each node u computes the minimum spanning tree with degree constraints in the graph $G_u(V_u, E_u)$ using the NDPSO algorithm independently in the previously section.

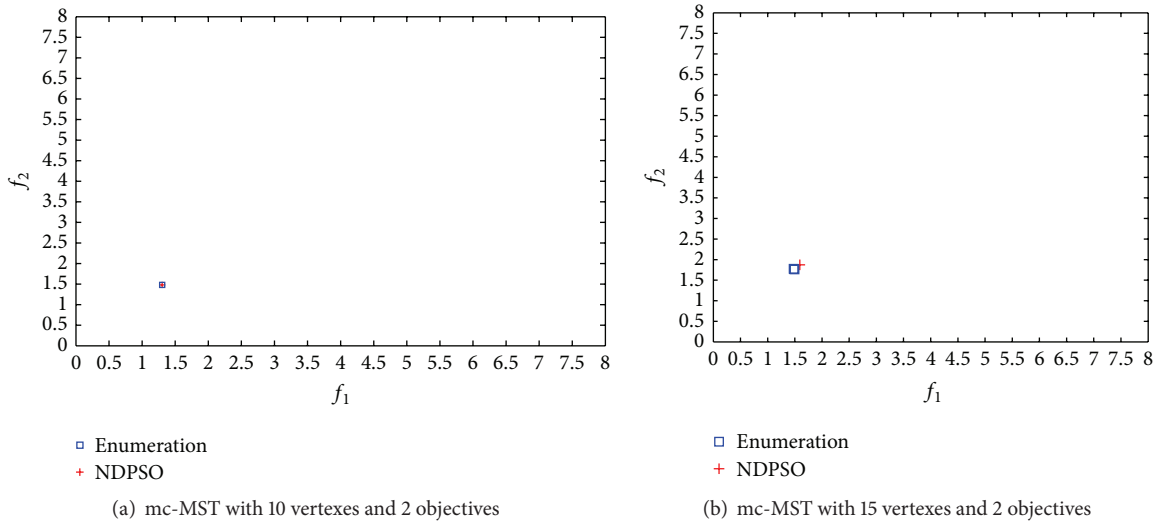
Step 3. Each node u has to adjust its transmission power according to its local minimum spanning tree $T_u(V(T), E(T))$, where $V(T) = V_u$, $E(T) \subset E_u$. At this time, node $u$requires a power lever that can reach the farthest one-hop neighbor in T_u.

5. Simulation Results

In this section, we first conduct simulations to compare the performance of PSO under our multiobjective strategy (MOPSO) with two classic multiobjective evolutionary algorithms in three standard test functions. Furthermore, the solution sets generated by NDPSO are also compared with solution sets from the enumerating algorithm [19] which is proved to find all Pareto optimal spanning trees. Finally, we conduct extensive simulations to compare the performance of our proposed MCMST scheme with other existing schemes.

5.1. Performance Analysis of MOPSO

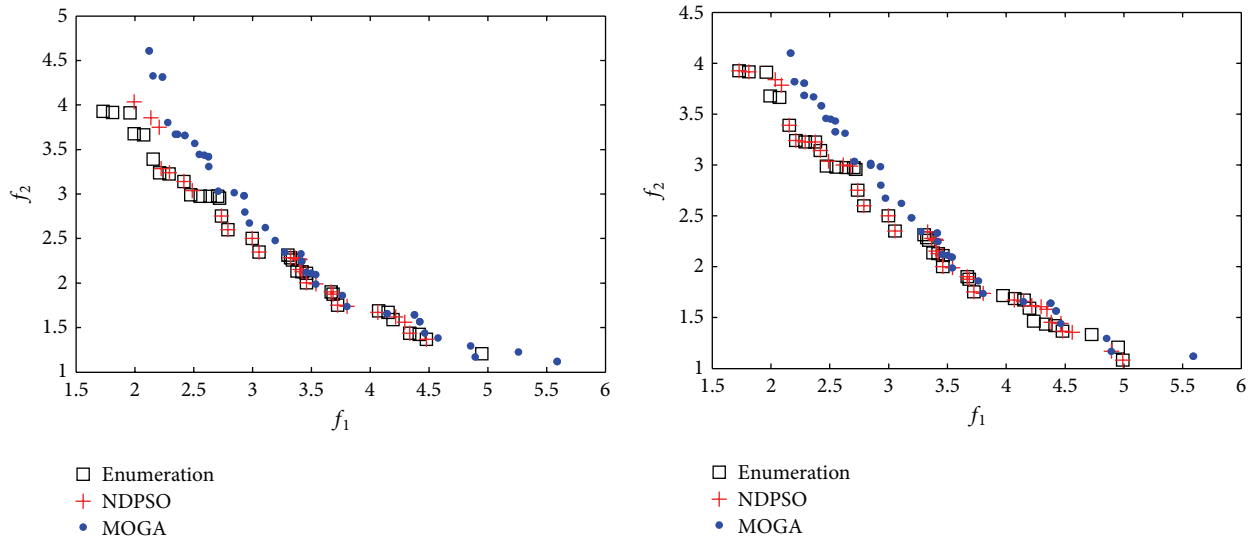
5.1.1. Experimental Setup. We use three standard test functions (ZDT1, ZDT2, and ZDT6) [38] and compare MOPSO with two classic multiobjective evolutionary algorithms



(a) mc-MST with 10 vertexes and 2 objectives

(b) mc-MST with 15 vertexes and 2 objectives

Figure 5: The effect of NDPSO with correlation = 0.95 on the different problem.



(a) mcd-MST with $d = 3$

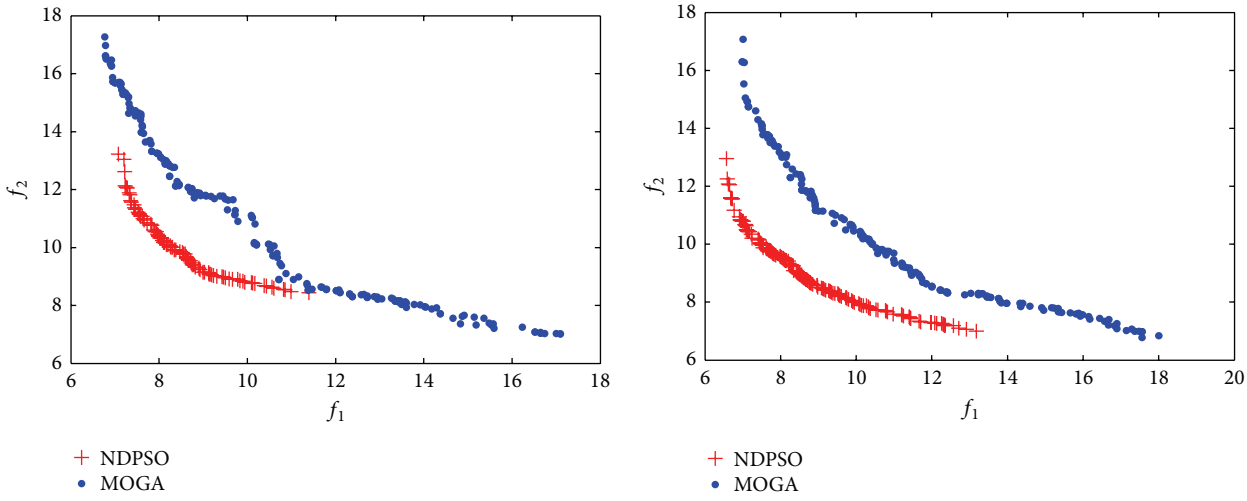
(b) mcd-MST with $d = 4$

Figure 6: Comparison on 15 vertices with $d = 3$ and $d = 4$.

(NSGA [39] and SPEA [40]). The three test functions represent some typical function types of noninferior front in multiobjective optimization problem, respectively. ZDT1 is a convex function, ZDT2 is a nonconvex function, and ZDT6 is noncontiguous function. In order to avoid algorithm randomness of a singular run, we independently run each algorithm for 20 times and compare the results of 20 experiments based on the statistical methods of hypothesis testing [41]. The experimental parameters are set as $c_1 = c_2 = 2.0$, $\sigma_s^1 = 0.01$, and $\sigma_s^2 = 1$. In addition, pop = 100 denotes the scale of population of all algorithms, and the maximum iteration times is 500 in each run.

To evaluate the performance of those multiobjective algorithms, Zitzler et al. developed several indicators [42]. Here, we adopt three quantitative indicators (unary additive EPS indicator $I_{\varepsilon+}^1$, HYP indicator I_H^-, and unary R2 indicator I_{R2}^1) in this paper.

5.1.2. Results and Analysis. As to the three test functions ZDT1, ZDT2, and ZDT6, in most cases, the proposed algorithm in this paper (MOPSO) can converge to the Pareto front within 100 iterations and have a small computational cost. Figure 1 shows the distribution situation of Pareto solution at a certain run. From the figure, we can infer that for each function, algorithm can obtain a well distribution of Pareto front. The results of detailed performance parameters are shown in Figure 2, and the Kruskal Wallis test results of statistical analysis are shown in Table 1.



(a) mcd-MST with $d = 3$ (b) mcd-MST with $d = 4$

FIGURE 7: Comparison on 50 vertices with $d = 3$ and $d = 4$.

From Figure 2 and Table 1, we can see that as to ZDT2 issue, MOPSO proposed in this paper is better than other multiobjective evolutionary algorithms in three indicators under the circumstance of same dominance rank distribution. So MOPSO algorithm is good at solving nonconvex Pareto front function. For ZDT6, MOPSO algorithm is superior to HSGA II and SPEA 2 on both R2 and HYP indicator, while MOPSO is little inferior to them on EPS indicator. According to [42], it also shows that if the results from multiple Pareto compliant indicators are different, it is hard to know which algorithm is better. It is obvious that MOPSO algorithm can obtain a better result in the front nonuniform ZDT6. As to ZDT1, although MOPSO is inferior to NSGA II and SPEA 2 on three parameters, the value of overall difference is small. Considering that it is unable to find manifest difference from other evolutionary algorithm on dominance rank distribution of solution and MOPSO can converge to Pareto front within few number of iteration times, MOPSO algorithm can obtain a satisfying solution in convex Pareto font optimization function. According to the results of three test function, we can see that MOPSO algorithm proposed in this paper has strong global search ability, and it can converge to the Pareto front by less computing cost. Furthermore, its distribution of solution is quite good. What we have mentioned previously demonstrates that MOPSO is worth being studied in the field of multiobjective optimization problem.

5.2. Performance Analysis of NDPSO

5.2.1. Experimental Setup. Let us consider a complete graph $G = (V, E)$, where $n = |V|$ and each edge has m weights, the problem's goal is to find all Pareto optimal solutions of G. The coordinate of vertexes in G and the weights of each edge are generated randomly. Suppose each edge in the graph has two weights, we propose generations for the following types:

(a) random: random (uncorrelated) real number weighted graphs;

(b) correlated: random correlated real number weighted graphs;

(c) anticorrelated: random anticorrelated real number weighted graphs.

In [18], Knowles proposed an algorithm to generate correlated (and anticorrelated) graphs. Knowles declared that all subsequent weights are either positively or negatively correlated with respect to the first component, and the values of the weights are within a same range. All the weights are of course not negative, but we find that the algorithm will generate negative value when $\alpha \leq 0$, and we presented an improved algorithm in [19] to generate correlated (and anticorrelated) graphs. The results of the improved algorithm can be seen in [19].

5.2.2. Results and Analysis. In [19], we proposed an improved enumeration algorithm which is proved to be able to find all the optimal solutions. So it can be a tool for evaluating the performances of other heuristic algorithms instead of the enumeration algorithm of Zhou and Gen [15]. In the first simulation, to evaluate the NDPSO algorithm, the solution sets generated by it are compared with solution sets from the improved enumeration [19], and the results are displayed from Figures 3, 4, and 5. Here, we consider 2-objective optimal problem. And the parameters are set as follows: the number of vertices is 10 and 15, respectively, population size is 50, and maximum generation is 1000.

From the results, we can draw a conclusion that the NDPSO algorithm has similar performance with increased size compared with the MOGA in [19]. However, the population size and the maximum generation in MOGA are set as 400 and 20000, respectively. Thus, the NDPSO is superior to MOGA in space and time complexity, and it is really effective to deal with the mc-MST problem.

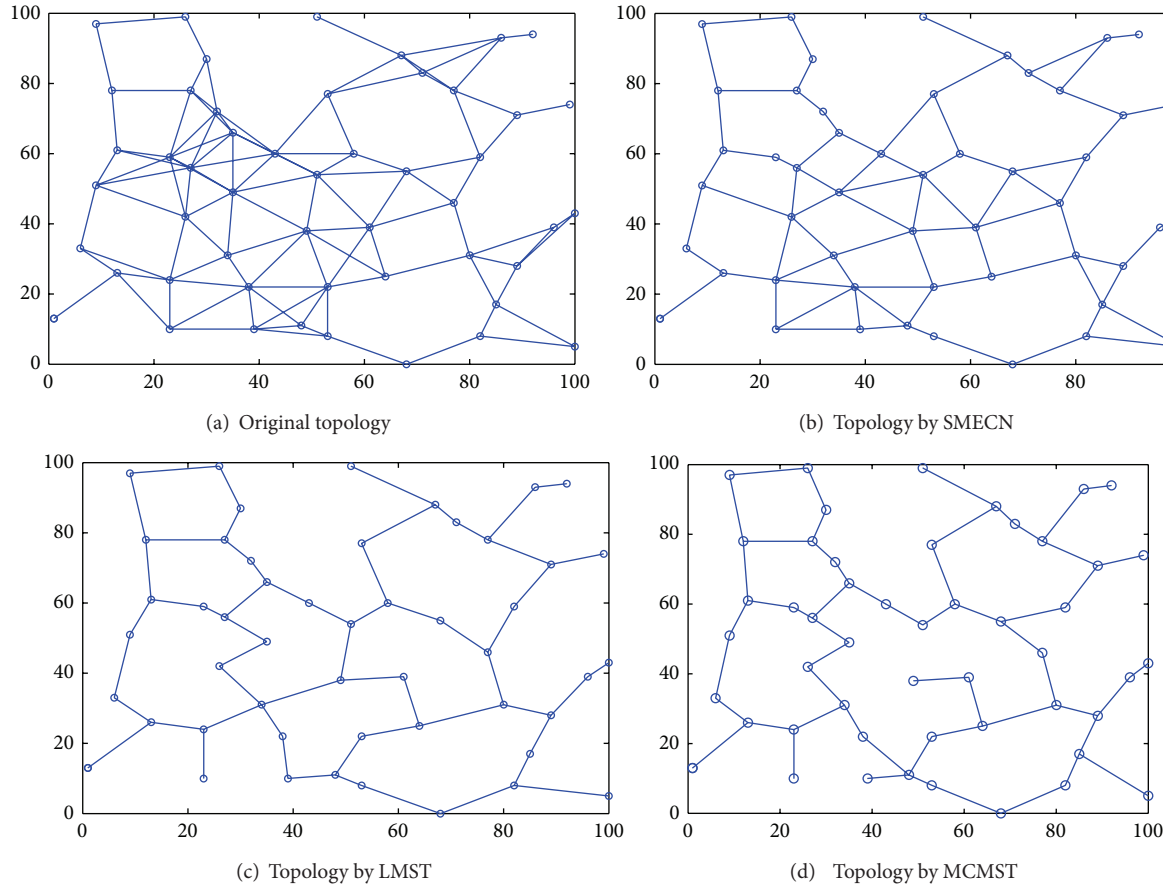


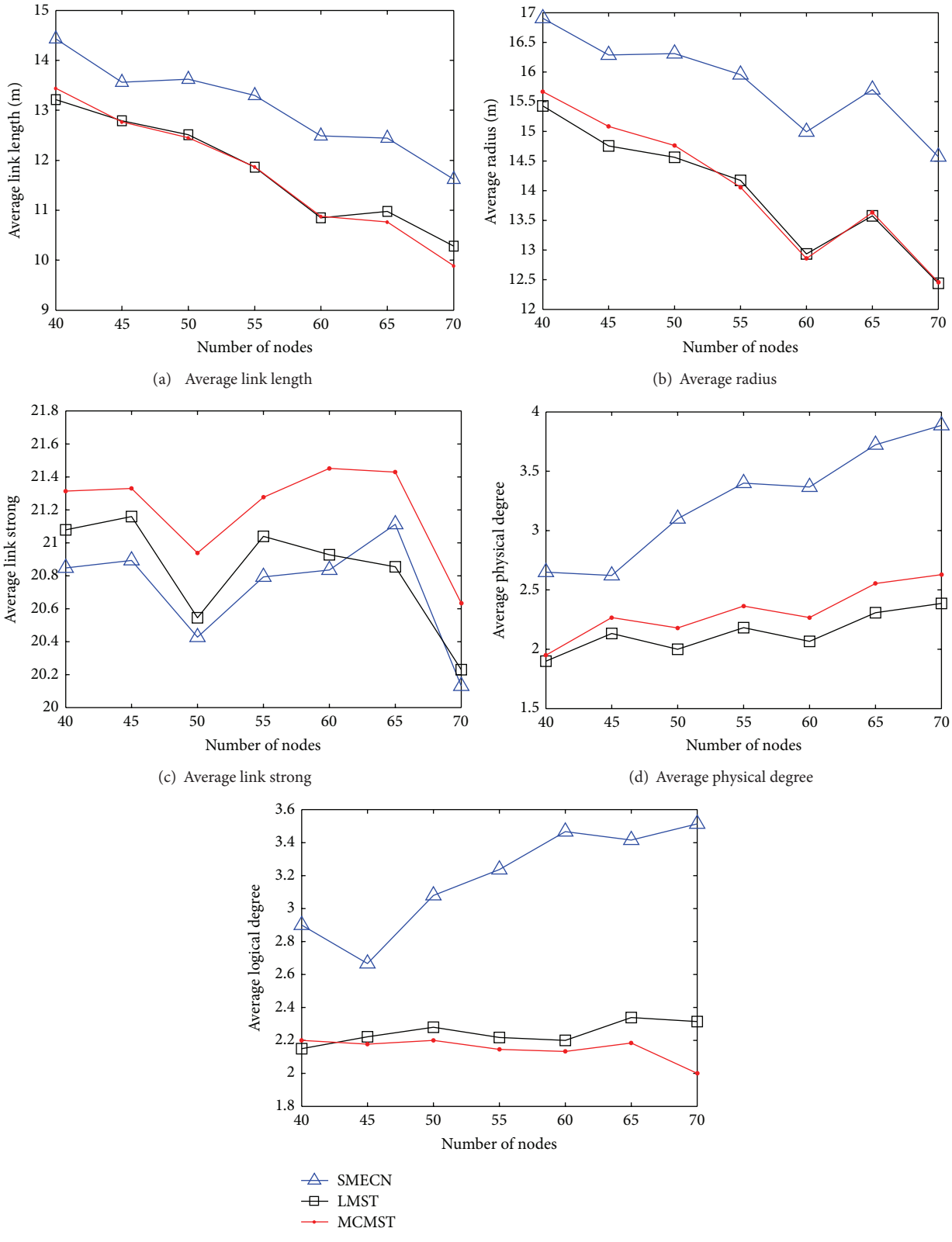
Figure 8: Network topologies derived under different topology control scheme.

In real network environmental, to satisfy the requirement of lifetime objective in wireless sensor networks and aim at the defect that high redundancy of connectivity or low robust of structure in traditional schemes, the model of topology control in WSNs can be transformed into a problem of mcd-MST. So in the second simulation, the problem's goal is to find mcd-MST solutions of G. The coordinate of vertexes in G and the weights of each edge are generated randomly. For simplicity, however, we suppose each edge in the graph has only two weights, which can be easily extended. We display the results from Figures 6 and 7. Figure 6 shows that the NDPSO algorithm converges to the real Pareto optimal solutions better than the MOGA in [19] under the same parameters setting no matter $d = 3$ or $d = 4$. From Figure 7, as a direct representation, we can find that the proposed NDPSO algorithm dominate the MOGA algorithm.

5.3. Performance Analysis of MCMST

5.3.1. Experimental Setup. Our simulation is done for a network of n nodes that are placed uniformly at random in a rectangular region of 100 by 100 meters. We assume that all nodes have the same maximum emission radius R, and the Euclidean distance between node i and node j is $d(i, j)$. The transmission power between node i and node j is defined as $C_{ij} = k * d(i, j)^2$, $k = 0.005$, the robustness of an edge between node i and node j is defined as $R_{ij} = e(i, j)/(e_i^2 + e_j^2)^{1/2}$, the maximal transmission range is $R_{\max} = 20$ m for all the nodes, and the initial energy of each node is equivalent to a random value between 20 and 40 Joule.

5.3.2. Result and Analysis. In the first simulation, 50 nodes are uniformly distributed in the region. Figure 8(a) shows the initial topology generated using the maximum transmission power, and the topology generated by different topology control schemes is shown in Figures 8(b), 8(c), and 8(d), respectively. As shown in Figure 8, SMECN, LMST and MCMST all dramatically reduce the average node degree while maintaining network connectivity. Moreover, LMST, and MCMST outperform SMECN in the sense that few edges formed in the final network topology, because they take the demand for low-power consumed into account and try to reduce nodes' transmission powers. However, the network topology generated by MCMST is slightly different form the one derived by LMST because LMST has not considered the robustness of topology structures, and at the same time MCMST has taken the edge coverage and node degree constrained into consideration.



(a) Average link length

(b) Average radius

(c) Average link strong

(d) Average physical degree

(e) Average logic degree

Figure 9: Performance comparisons among different topology control scheme.

Table 1: The Kruskal Wallis test results of EPS, HYP, and R2 indicator.

		MOPSO	NSGA II	SPEA 2
		ZDT 1		
$I^1_{\varepsilon+}$	MOPSO	—	1	1
	NSGA II	$4.16519E-17$	—	1
	SPEA 2	$3.48666E-34$	$4.16519E-17$	—
I^-_H	MOPSO	—	1	1
	NSGA II	$2.53616E-16$	—	1
	SPEA 2	$4.00793E-32$	$4.88893E-15$	—
I^1_{R2}	MOPSO	—	1	0.99996
	NSGA II	$7.6029E-08$	—	0.0551455
	SPEA 2	$3.99655E-05$	0.944854	—
		ZDT 2		
$I^1_{\varepsilon+}$	MOPSO	—	$2.20462E-06$	$2.16472E-05$
	NSGA II	0.999998	—	0.72699
	SPEA 2	0.999978	0.27301	—
I^-_H	MOPSO	—	0.000234484	0.00112884
	NSGA II	0.999766	—	0.689058
	SPEA 2	0.998871	0.310942	—
I^1_{R2}	MOPSO	—	$1.18542E-14$	$7.16123E-14$
	NSGA II	1	—	0.658011
	SPEA 2	1	0.341989	—
		ZDT 6		
$I^1_{\varepsilon+}$	MOPSO	—	1	1
	NSGA II	$3.27923E-07$	—	1
	SPEA 2	$1.28526E-20$	$3.94864E-11$	—
I^-_H	MOPSO	—	$3.10337E-14$	$1.21161E-20$
	NSGA II	1	—	0.000456755
	SPEA 2	1	0.999543	—
I^1_{R2}	MOPSO	—	$8.72107E-20$	$4.22686E-29$
	NSGA II	1	—	$1.43107E-07$
	SPEA 2	1	1	—

This table shows each pair of the value of P which is composed of algorithm O_R (row) and algorithm O_C (column), and the corresponding alternative hypothesis is that algorithm O_R is superior to algorithm O_C in indicator. The value of significance level is 0.05.

In the second simulation, we vary the number of nodes in the region from 40 to 70. The average radius and the average link length for the topologies generated using the SMECN, LMST, and MCMST with link removal are shown, respectively, in Figures 9(a) and 9(b). MCMST and LMST outperform SMECN in the both two cases, and MCMST scheme has similar performance with LMST scheme. However, with the size of nodes increasing, MCMST has better performances than LMST in terms of the average link length. The results imply that our proposed scheme can provide a better spatial reuse and is power-efficient. As shown in Figure 9(c), the average link strong for the topologies generated using our scheme outperforms than that of SMECN and LMST because the structure robustness of network is considered in our proposed MCMST scheme. The average physical degree and the average logic degree for the topologies using SMECN, LMST, and MCMST are shown, respectively, in Figures 9(d) and 9(e). From Figure 9(d), we can know that MCMST, and LMST outperform SMECN and MCMST is little inferior to LMST. As shown in Figure 9(e), the average logical degree for the topologies generated using the MCMST has better performance than LMST and SMECN with the size of nodes increasing. It implies that our proposed scheme can get a better network topology with low radio interference.

In a word, MCMST scheme proposed in this paper has made a tradeoff among the energy consumption, radio interference, and structure robustness, and the performance of the resulting network topology has been greatly improved.

6. Conclusions

Due to the demand for the optimization of the network lifetime, this paper adopts the idea of a local minimum spanning tree, transforms the model into a multicriteria degreeconstrained minimum spanning tree, and proposes a NDPSO algorithm. The phenotype sharing function of the objective space is applied in the definition of fitness function to obtain a better approximation of true Pareto front, and

the global convergence of the algorithm is proved using the theorem of Markov chain. Then, a topology control scheme based on NDPSO is put forward. Simulation results show that the obtained topology in this paper has low overall power consumption, high structural robustness, and controllable internode communication interference characteristics and can effectively prolong the lifetime of WSNs. And our algorithm can obtain a better approximation of true Pareto front.

Future work will cover the following problem of the proposed MCMST scheme: (1) consider the reliability in wireless sensor networks effectively; (2) how to reduce the complexity and improve the convergence efficiency of our NDPSO.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant no. 61103175 and no. 61103194, the Key Project of Chinese Ministry of Education under Grant no. 212086, the Technology Innovation Platform Project of Fujian Province under Grant no. 2009J1007, the Key Project Development Foundation of Education Committee of Fujian province under Grant no. JA11011, and the Fujian Province High School Science Fund for Distinguished Young Scholars under Grant no. JA12016.

References

[1] T. Laukkarinen, J. Suhonen, and M. Hannikainen, "A survey of wireless sensor network abstraction for application development," *International Journal of Distributed Sensor Networks*, vol. 2012, Article ID 740268, 12 pages, 2012.

[2] N. Ababneh, "Performance evaluation of a topology control algorithm for wireless sensor networks," *International Journal of Distributed Sensor Networks*, vol. 2010, Article ID 671385, 16 pages, 2010.

[3] L. Lobello and E. Toscano, "An adaptive approach to topology management in large and dense real-time wireless sensor networks," *IEEE Transactions on Industrial Informatics*, vol. 5, no. 3, pp. 314–324, 2009.

[4] S. Zarifzadeh, N. Yazdani, and A. Nayyeri, "Energy-efficient topology control in wireless ad hoc networks with selfish nodes," *Computer Networks*, vol. 56, no. 2, pp. 902–914, 2012.

[5] 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.

[6] Y. Ding, C. Wang, and L. Xiao, "An adaptive partitioning scheme for sleep scheduling and topology control in wireless sensor networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 20, no. 9, pp. 1352–1365, 2009.

[7] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *Proceedings of the 33rd Annual Hawaii International Conference on System Siences (HICSS-33 '00)*, pp. 1–10, January 2000.

[8] S. Lindsay and C. Raghavendra, "PEGASIS: power-efficient gathering in sensor information systems," in *Proceedings of IEEE Aerospace Conference*, pp. 1125–1130, 2002.

[9] L. Li, J. Y. Halpern, P. Bahl, Y. M. Wang, and R. Wattenhofer, "Analysis of a cone-based distributed topology control algorithm for wireless multi-hop networks," in *Proceedings of the 20th Annual ACM Symposium on Principles of Distributed Computing*, pp. 264–273, August 2001.

[10] V. Rodoplu and T. H. Meng, "Minimum energy mobile wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 8, pp. 1333–1344, 1999.

[11] L. Li and J. Y. Halpern, "A minimum-energy path-preserving topology-control algorithm," *IEEE Transactions on Wireless Communications*, vol. 3, no. 3, pp. 910–921, 2004.

[12] N. Li, J. C. Hou, and L. Sha, "Design and analysis of an mst-based topology control algorithm," in *Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communication Scocieties*, pp. 1702–1712, 2003.

[13] W. Z. Guo, J. H. Park, L. T. Yang, A. V. Vasilakos, N. X. Xiong, and G. L. Chen, "Design and analysis of a MST-based topology control scheme with PSO for wireless sensor networks," in *Proceedings of IEEE Asia-Pacific Services Computing Conference*, pp. 360–367, December 2011.

[14] Y. H. Chen, "Polynomial time approximation schemes for the constrained minimum spanning tree problem," *Journal of Applied Mathematics*, vol. 2012, Article ID 394721, 8 pages, 2012.

[15] G. Zhou and M. Gen, "Genetic algorithm approach on multi-criteria minimum spanning tree problem," *European Journal of Operational Research*, vol. 114, no. 1, pp. 141–152, 1999.

[16] J. Gottlieb, B. A. Julstrom, G. R. Raidl, and F. Rothlauf, "Prufer numbers: a poor representation of spanning trees for evolutionary search," in *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 343–350, 2001.

[17] N. Srinivas and K. Deb, "Multi-objective optimization using nondominated sorting in genetic algorithms," *Evolutional Computation*, vol. 2, no. 3, pp. 221–248, 1994.

[18] J. D. Knowles, *Local-search and hybrid evolutionary algorithms for Pareto optimization [thesis]*, University of Reading, West Berkshire, UK, 2002.

[19] G. Chen, S. Chen, W. Guo, and H. Chen, "The multi-criteria minimum spanning tree problem based genetic algorithm," *Information Sciences*, vol. 177, no. 22, pp. 5050–5063, 2007.

[20] D. A. van Veldhuizen and G. B. Lamont, "Multiobjective evolutionary algorithm test suites," in *Proceedings of the 14th ACM Symposium on Applied Computing (SAC '99)*, pp. 351–357, March 1999.

[21] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, vol. 4, pp. 1942–1948, December 1995.

[22] X. Li and X. Yao, "Cooperatively coevolving particle swarms for large scale optimization," *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 2, pp. 210–224, 2012.

[23] S. S. Jiang, Z. W. Zhao, S. Mou, Z. S. Wu, and Y. Luo, "Linear decision fusion under the control of constrained PSO for WSNs," *International Journal of Distributed Sensor Networks*, vol. 2012, Article ID 871596, 11 pages, 2012.

[24] Y. Morsly, N. Aouf, M. S. Djouadi, and M. Richardson, "Particle swarm optimization inspired probability algorithm for optimal camera network placement," *IEEE Sensors Journal*, vol. 12, no. 5, pp. 1402–1412, 2012.

[25] Y. Shen, G. Wang, and C. Tao, "Particle swarm optimization with novel processing strategy and its application," *International Journal of Computational Intelligence Systems*, vol. 4, no. 1, pp. 100–111, 2011.

[26] D. Caputo, F. Grimaccia, M. Mussetta, and R. E. Zich, "Genetical swarm optimization of multihop routes in wireless sensor networks," *Applied Computational Intelligence and Soft Computing*, vol. 2010, Article ID 523943, 14 pages, 2010.

[27] K. Zielinski, P. Weitkemper, R. Laur, and K. D. Kammeyer, "Optimization of power allocation for interference cancellation with particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 1, pp. 128–150, 2009.

[28] R. V. Kulkarni and G. K. Venayagamoorthy, "Particle swarm optimization in wireless-sensor networks: a brief survey," *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, vol. 41, no. 2, pp. 262–267, 2011.

[29] W. Z. Guo, H. L. Gao, G. L. Chen, and L. Yu, "Particle swarm optimization for the degree-constrained MST problem in WSN topology control," in *Proceedings of International Conference on Machine Learning and Cybernetics*, vol. 3, pp. 1793–1798, July 2009.

[30] W. N. Chen, J. Zhang, H. S. H. Chung, W. L. Zhong, W. G. Wu, and Y. H. Shi, "A novel set-based particle swarm optimization method for discrete optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 2, pp. 278–300, 2010.

[31] G. Lapizco-Encinas, C. Kingsford, and J. Reggia, "Particle swarm optimization for multimodal combinatorial problems and its application to protein design," in *Proceedings of IEEE Congress on Evolutionary Computation*, pp. 1–8, July 2010.

[32] Q. K. Pan, M. F. Tasgetiren, and Y. C. Liang, "A discrete particle swarm optimization algorithm for the permutation flowshop sequecing problem with Makespan criteria," in *Proceedings of the 26th SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence*, pp. 19–31, 2006.

[33] S. H. Ling, F. Jiang, H. T. Nguyen, and K. Y. Chan, "Hybrid fuzzy logic-based particle swarm optimization for flow shop scheduling problem," *International Journal of Computational Intelligence and Applications*, vol. 10, no. 3, pp. 335–356, 2011.

[34] R. M. Aliguliyev, "Clustering techniques and discrete particle swarm optimization algorithm for multi-document summarization," *Computational Intelligence*, vol. 26, no. 4, pp. 420–448, 2010.

[35] W. Z. Guo, N. X. Xiong, A. V. Vasilakos, G. L. Chen, and C. L. Yu, "Distributed k-connected fault-tolerant topology control algorithms with PSO in future autonomic sensor systems," *International Journal of Sensor Networks*, vol. 12, no. 1, pp. 53–62, 2012.

[36] R. Balling, "The maximin fitness function: multi-objective city and regional planning," in *Proceedings of the 2nd International Conference on Evolutionary Multi-Criterion Optimization*, vol. 2632, pp. 1–15, 2003.

[37] F. Neumann and M. Laumanns, "Speeding up approximation algorithms for NP-hard spanning forest problems by multi-objective optimization," in *Electronic Colloquium on Computational Complexity*, Report no. 29, 2005.

[38] E. Zitzler, *Evolutionary Algorithms for Multi-Objective Optimization: Methods and Applications*, Swiss Federal Institute of Technology, Zurich, Switzerland, 1999.

[39] R. E. Steuer, *Multiple Criteria Optimization: Theory, Computation, and Application*, John & Wiley Sons, New York, NY, USA, 1986.

[40] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: improving the strength pareto evolutionary algorithm," in *Proceedings of Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems (EUROGEN '01)*, September 2001.

[41] W. J. Conover, *Practical Nonparametric Statistics*, John Wiley & Sons, New York, NY, USA, 3rd edition, 1999.

[42] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. da Fonseca, "Performance assessment of multiobjective optimizers: an analysis and review," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 2, pp. 117–132, 2003.

Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2013, Article ID 493678, 11 pages
http://dx.doi.org/10.1155/2013/493678

Research Article

A Fault-Tolerant Method for Enhancing Reliability of Services Composition Application in WSNs Based on BPEL

Zhao Wu,[1] NaiXue Xiong,[2] Wenlin Han,[3] Yan N. Huang,[1] Chun Y. Hu,[1] Qiong Gu,[1] and Bo Hang[1]

[1] *School of Mathematics and Computer Science, Hubei University of Arts and Science, Xiangyang 441053, China*
[2] *School of Computer Science, Colorado Technical University, Colorado Springs, CO 80907, USA*
[3] *Department of Computer and Science, University of Alabama, Tuscaloosa, AL 35487, USA*

Correspondence should be addressed to NaiXue Xiong; nxiong@coloradotech.edu

Received 4 February 2013; Accepted 22 February 2013

Academic Editor: Hongju Cheng

In recent years, some approaches have been presented for the seamless integration of WSNs with the existing, widely deployed SOA technologies such as XML, Web services, and the Business Process Execution Language (BPEL) to build a wireless sensor networks service application. However, there a great challenge on fault tolerant in WSNs. In this paper, we present our framework and approach to enhance the reliability of service composition applications in WSNs through modeling and analyzing a wireless sensor networks service application based on BPEL with exception handler and compensation mechanism. At first, we analyze all possible states during the execution of BPEL instance in WSNs. Then, we present a state framework for modeling execution context in BPEL instance in WSNs. Based on this framework, we analyze state transition and operational semantics in the case of both correct execution and exceptional execution of BPEL instance in WSNs. Furthermore, we propose the state transition models for three types of activities in BPEL instance. In the end, we present a formal approach to model the execution context in BPEL for WSNs. Using this formal model, one can describe and analyze the control flow result from the exception handler and compensation mechanism in BPEL instance for WSNs.

1. Introduction

Despite the amount of research targeted at middleware systems for wireless sensor networks (WSNs), they are still not widely used in industry. Certainly, one major issue is the different programming methodology. While WSNs are optimized for low-power, low-cost, and a small form factor, Enterprise-IT systems are typically equipped with more resource and are connected to the power grid. In Enterprise IT, it is significant to adapt business processes and the underlying software infrastructure quickly and flexibly to react to changes on the markets. To achieve this goal, organizations focus on modeling, analysis, and adaptation of business processes since the early 1990s. With the advent of service-oriented architecture (SOA) based on Internet standards, more and more businesses are transferred to this architecture.

Parallel to this development, WSNs are envisioned to become an integral part of the Future Internet where they extend the Internet to the physical world. Combined with each other, these two trends lay the groundwork for a new class of applications where all kinds of devices ranging from simple sensor nodes to large-scale application servers interact to drive business processes which were not possible before. That way, data stemming from a WSN may influence the control flow of a business process in real time or even trigger a business process. To achieve this level of integration, WSNs must seamlessly interoperate with the existing widely deployed SOA technologies such as XML, web Services, and the Business Process Execution Language (BPEL) to name only a few. In recent years, some approaches have been presented for the seamless integration WSNs with these SOA technologies to build a wireless sensor networks service application successfully. In these approaches, WSNs

are packaged as some standard Web services which can be published, located, and invoked across the Web. Therefore, based on BPEL, these WSNs services can be combined into a workflow to fulfill some certain tasks in a web services composition (WSC) way.

As standard Web services, the capsulated WSNs are of all characters of Web services. Web services provide the basis for the development and execution of business processes that are distributed over the Internet and are available via standard interfaces and protocols [1–3]. Web services have the characters of interoperation, platform independent, self-described, and loose coupling [4]. Web service composition (WSC) is one of the most promising ideas underlying Web services: new functionalities can be defined and implemented by combing and interacting with the preexisting Web services. WSC refers to the process of composing multiple stateless atomic Web services into statefull complex applications [5–8].

Under satisfying the precondition of functional demands, the fault tolerant is the key for a wireless sensor networks service application. However, deploying a test-bed system to evaluate WSNs application system is very expensive and time consuming. Therefore, the performance modeling and analysis of WSC in WSNs is the important research direction of business process reengineering [9–17].

In our previous work [18, 19], we studied the performance modeling and analysis methods for the basic control flow of WSC under the circumstance that BPEL instance is executed correctly. We presented a novel performance simulation model for WSC, called STPM+. The STPM+ model can support modeling and simulating the time and nontime QoS metrics of WSC. Based on the stochastic timed colored Petri net, the STPM+ model can simulate and predict multiple QoS metrics such as cost, reliability, and credibility. We designed and realized a visual performance simulation tool, called VisualWSCPE. One can simulate and analyze the execution of BPEL instance, assess its performance, and find out the potential performance bottlenecks with VisualWSCPE. However the transaction property of WSC in WSNs is not considered during we fulfilled the above modeling and analysis. This means that the execution context in BPEL cannot be modeled and analyzed based on the above model. So, the modeling and analysis for the transaction and exception handling of BPEL instance in WSNs should be studied further.

Transaction is a very important concept related to exception handling and compensation [18, 19]. Transaction constitutes a single set of the logical operation unit. All operations should be completed successfully, or fail totally and roll back to the previous state before the transaction is executed [20, 21]. In order to ensure its integrity, transaction should have four properties: atomicity, consistency, isolation, and durability, which are called ACID [22, 23].

Business processes often need to use the concept of transaction to handle exception and compensation. The use of ACID transactions is usually limited to local updates because of trust issues in a business process. Because the invocation of Web service cannot usually lock the resources across different enterprises, the common transaction mechanism, for example, two-phase commit, cannot be used to handle this circumstance. Besides, the locks and isolation cannot be maintained for a long period during occurrence of the technical and business errors. Consequently, the demand of long-running transactions (LRT) is put forward [20]. LRT refers to a longer duration transaction, which cannot achieve the restoration of data and state through the common rollback mechanism. Because the cross-organizational resources cannot be locked, even if the invoked atomic Web services under LRT all meet the requirements of ACID, the overall LRT cannot still satisfy the requirements of ACID [24, 25]. Generally, when an LRT fails, the simply rollback will be used to undo the effects resulted from these executed operations. A reverse process is used to perform these rollback operations, which is called "compensation" [26, 27]. Compensation is a particular approach to the business data management. So, it is always a part of business logics. The roll of compensation in LRT is different from that of atomic rollback operations provided by database management system for ACID of transaction. The roll of compensation in LRT is reflected in the improved security. For example, in the Internet, one may lock a company's data by performing an invocation operation to a certain Web service. The invocation operation in LRT might result in the denial of service attacks in the case of traditional transaction handling. Compensation in LRT can avoid this problem. The use of compensation means that data will not be locked in a long duration. So, the business data cannot be locked in a long period time and the denial of service will never happen. However, the use of compensation also causes a new problem simultaneously. It results in the fact that the LRT loses ACID prosperities and the isolation cannot be at least satisfied. It is because the business data is visible in the whole transition duration from the initial update operation to compensation.

BPEL provides some compensation mechanisms by providing the ability for flexible control of the reverse operation. In BPEL, the fault handling and compensation can be defined to support LRT in an application-specific manner [28, 29]. Compensation operations are triggered by the occurrence of exception event in BPEL specification [30].

To verify the availability of exception handling and compensation mechanism of BPEL instance in WSNs, we validate the modeling and analysis methods for the execution context of BPEL in this paper. Our contributions are threefold.

(i) We present a state framework of WSC for WSNs that is based on BPEL. Based on this framework, we can analyze and construct state transition model for all kinds of BPEL activities in WSNs.

(ii) For three kinds of activities in BPEL: *basic activity*, *structural activity*, and *scope activity*, we analyze various state transitions in terms of respective mechanisms of exception handling and compensation in WSNs. And we present the state transition models for each activity in BPEL. Based on our state transition models, one can build the state transition system of WSC for WSNs.

(iii) We present an approach to model the state transition process with state transition system for WSNs. Using the approach mentioned above, one can describe and

verify the control flows resulted from exception handler and compensation mechanism of BPEL instance for WSNs.

This paper is organized as follows. The next section introduces the related works on modeling and analysis of execution context in BPEL instance for WSNs. In Section 3, we analyze all possible states in BPEL instance for WSNs at first. Then we present a state framework for the execution context in BPEL by which we can analyze exception states and compensation states of BPEL activities. In Section 4, we analyze the state transitions of three types of activities of BPEL based on the state framework for WSNs at first. Then we present their state transition models, respectively. In the end, we present a formal approach to model the execution context of BPEL for WSNs based on the state transition model in Section 5.

2. Related Works

This paper focuses on the modeling and analysis of the execution context of BPEL. There have been several formal models presented in the literature. Kazhamiakin et al. [31] made researches on the communication behaviors of different participants in BPEL and presented an approach to model such behaviors. It describes the different states in course of execution of BPEL activity and the process by which the system evolves to a new state resulting from executing some actions. In addition, Kovács et al. [23] proposed a formalism method for capturing the behaviors of business process implemented in BPEL.

It uses transition system as the formal model of workflow and presents the state transition models of different activities. Furthermore, Nakajima [32] presented a model based on the extended finite automation to analyze the instance of BPEL, which maps every control flow into individual extended finite automation, so every activity can be modeled into different extended finite automation.

However, there are some limitations in the above researches. Because the above studies mainly concentrate on the verification of the static model of the instance of BPEL, the temporal behaviors are not considered during the execution of BPEL instance. Besides, the state transition model presented in [18] only describes the communication behavior of business process. The restriction condition of state transition is not considered. So, it is not sufficiently complex to model and simulate the dynamic execution of the instance of BPEL. In addition, the state transition system proposed in [31] presents the state transition model of basic activity and structural activity, but these models are too simple to cover all states in the course of execution of structural activity. And these models do not consider the restriction condition of state transition. So, they are unable to describe the behaviors and states during the execution of the instance of BPEL. In the end, the model proposed in [23] does not consider event, fault handling and compensation, so it cannot obtain all states and behaviors during the execution of the instance of BPEL.

To overcome the above shortcomings, we analyze all possible states in the execution of the instance of BPEL at first. Then, we present a sufficient state frame model to capture the communication behavior and all states of BPEL instance. Based on our state frame model, we construct a state transition system for modeling the execution context of BPEL instance. And we analyze the operational semantics of state transitions.

3. State Analysis of the Activities in BPEL

Web services that are modeled by WSDL are stateless. It means that the states of Web services cannot be captured and maintained. BPEL can describe the invocation relationships among Web services components within a business process. So, it can be used for modeling a WSC instance. Moreover, BPEL provides the context of behaviors of each activity including fault handlers, event handlers, compensation handlers, data variables, and correlation sets. When an activity instance is finished, the next activity instance to be executed is selected and its state is changed to *Ready*. After this, the instance of activity may go through a number of internal states. Finally, if all the associated processing has been performed successfully, its state is changed to *Completed*. These two states are crucial to control flow. And any formal semantics of control flow constructs has to take at least these two states into account explicitly.

In BPEL, the LRT mechanism is supported by an essential concept *scope*. We can regard *scope* as a kind of special activity which can include other activities. All the activities in a *scope* have three running modes.

Mode 1. When a <*scope*> activity starts running, it will execute in a normal way. So, all the activities and event handlings within the <*scope*> will be executed normally.

Mode 2. During the execution of <*scope*> activity, once a certain activity instance has a fault, the running mode of <*scope*> activity will change from normal to fault handling. In consequence, all the activities and event handlings will receive the termination message to be stopped.

Mode 3. After all the activities and event handlings are stopped successfully, the <*scope*> activity will execute compensation handling. In this case, the compensation handling will remove the effects of fulfilled activities within the <*scope*> activity and restore them to the former state. Figure 1 shows the compensation mechanism.

Besides the <*scope*> activity, BPEL also provides a structure, called *control link*, and two kinds of labels related to activity, called *join condition* and *transition condition*, which are used for the definition of priority level, synchronization, and conditional dependency. BPEL provides a mechanism, called dead path elimination, to prevent the emergence of death lock. Suppose that there is a *control link* between activity A and activity B. Then, it indicates that B cannot start running before A has fulfilled or has skipped. B can start running only when the *join condition* is true. Otherwise,

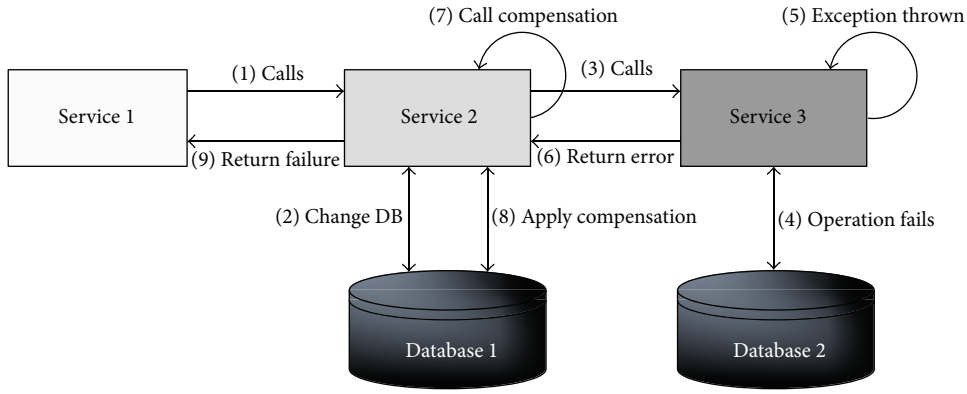


Figure 1: The compensation mechanism in long-running transition.

B would be skipped. Thus, activity should have a *skipped state* to support the dead path elimination mechanism.

A composite Web services must go through several internal states from *ready* state to *completed* state. It is obvious that the running state hides in the internal states of composite web service. Due to the conditional dependency, the composite Web services must wait for the end of execution of other services and then it can be instanced and executed. Thus, the parent service of the composite Web services is blocked which indicates that the state of the parent service is a *blocked state*.

Based on the above analysis, we consider that a composite Web services owns seven states as follows.

(1) *Skipped*. BPEL provides a construct known as *control links* which, together with the associated notions of *join condition* and *transition condition*, supports the definition of precedence, synchronization, and conditional dependencies. A *control link* between activities A and B indicates that B cannot start before A has either completed or has been "skipped." Moreover, B can only be executed if its associated *join condition* evaluates to true; otherwise, B is skipped.

(2) *Ready*. Composite Web service is initializing.

(3) *Running*. Composite Web service has finished initialization and is running.

(4) *Blocked*. Due to synchronization, a composite Web service needs to wait for the completion of other composite Web services.

(5) *Failed*. Composite Web service has the fault during its running.

(6) *Terminated*. Composite Web service completes unsuccessfully or is terminated by context.

(7) *Completed*. Composite Web service performs successfully.

Figure 2 shows our state framework for a composite web service. Based on this state framework, we can build state transition model for all kinds of BPEL activities.

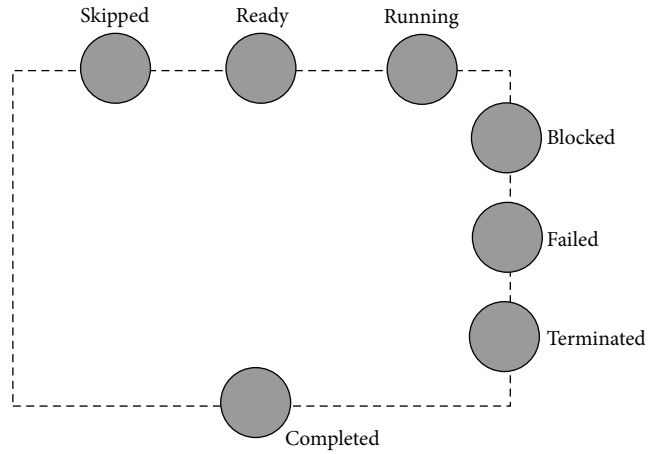


Figure 2: The state framework.

4. State Transition Model of Activities in BPEL

There are three kinds of activities in BPEL: basic activity, structural activity, and *scope* activity. We analyze the state transitions in terms of their respective mechanisms of exception handling and compensation handling.

4.1. State Transition Model of Basic Activity. Given that the initial state of a basic activity x is *Ready*, which indicates that its initialization has not completed. If x has a synchronization dependency on y and its join condition is false through evaluating this condition, and suppress join failure prosperity of BPEL instance is set to true, x will be skipped and will not be performed. Therefore, its state will be changed to *Skipped* from *Ready*, or x will complete initialization and its state will be also changed from *Ready* to *Running*. Because the main role of basic activity is the data processing and communications between other activities, *Failed* state of the basic activity is related to itself, for example, the failure of network communication or database operation. If the fault has occurred, basic activity x will capture and throw this exception, and its state will be also changed from *Running* to *Failed*. Moreover, these exceptions always register their

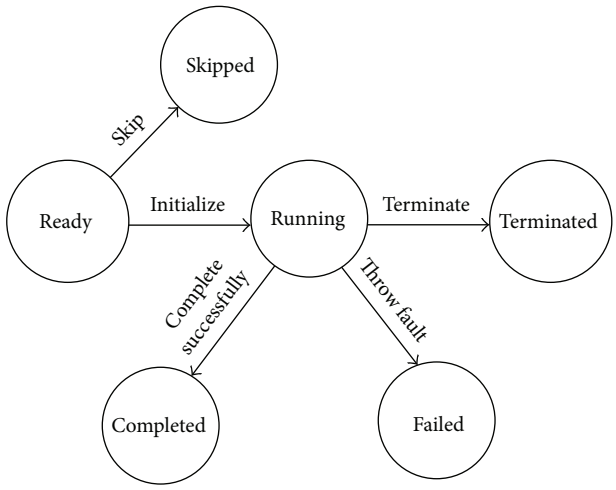
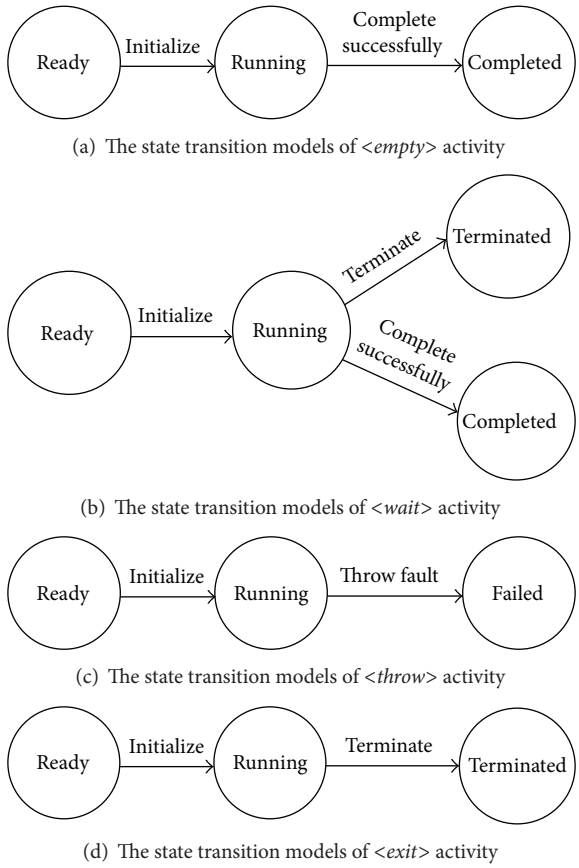


Figure 3: The state transition model of common basic.



(a) The state transition models of <*empty*> activity

(b) The state transition models of <*wait*> activity

(c) The state transition models of <*throw*> activity

(d) The state transition models of <*exit*> activity

Figure 4: The state transition models of special basic activities.

immediate *scope* activity as fault event, the corresponding *scope* activity will execute the fault handler, and it can forward termination event to the executing inner activities. Therefore, its state will be also changed from *Running* to *Terminated*. If x has completed successfully, its state will be also changed from *Running* to *Completed*. In consequence, Figure 3 describes the state transition model of common basic activity.

However, there are several special basic activities, for example, <*empty*> activity, <*wait*> activity, <*throw*> activity, and <*exit*> activity. Their state transition models are different from others. <*empty*> activity cannot be terminated or skipped. It cannot also own failed state because it does not perform any operators. Therefore, <*empty*> activity owns *Ready*, *Running*, and *Completed* as shown in Figure 4(a).

According to BPEL specification, <*wait*> activity cannot be skipped, because it does not own *Failed* state as shown in Figure 4(b).

The <*throw*> activity will explicitly throw exceptional event to its immediate <*scope*> activity. Generally speaking, this type of activity does not have a *Completed* state. And after throwing fault, it directly sets its state as *Failed*.

Similarly, according to BPEL specification, <*throw*> activity cannot be skipped and terminated. Therefore, it does not own *Skipped* and *Terminated* states. Its state transition model is described in Figure 4(c).

The <*exit*> activity will cancel the whole process instance. Its execution will trigger a termination event to Process. So when it has completed, its state will be changed to *Terminated*. Its state transition model is described in Figure 4(d).

The state transition of basic activity is triggered by the message events sent to its execution context, such as termination event or fault event. In general, the execution context of basic activity will receive the messages from its ancestor activities. Consequently, these messages will trigger the change of its state.

4.2. State Transition Model of Structural Activity. There are several structural activities in BPEL such as <*flow*> activity, <*sequence*> activity, <*if*> activity, <*while*> activity, <*repeatuntil*> activity, serial <*foreach*> activity, and <*foreach*> activity.

In this subsection, we mainly analyze the state transitions of these structural activities.

The <*flow*> activity provides concurrency and synchronization. Given a <*flow*> activity x, its initial state is *Ready*, which indicates that it has not finished initialization. If any branch of x has finished initialization; that is, the state of an inner activity within x is *Running*, the state of x is also changed from *Ready* to *Running*. If the state of x is *Running*, only when the states of all inner activities are changed to *Completed*, the state of x is changed from *Running* to *Completed*. If any inner activity of x throws the exception; that is, its state is *Failed*, the state of x is also changed to *Failed*, and x will send termination event for other inner activities, and if x receives termination event, it will also forward this event for its inner activities, only when all inner activities of x are *Terminated* successfully, and the state of x is changed to *Terminated*. If the state of x is *Ready* before finishing the initialization, and x has the synchronization dependencies on other activities, and its <joinCondition> is evaluated to false, its state is changed from *Ready* to *Skipped*. If the state of x is *Running*, and when the state of its all inner activities is *Blocked*, the state of x will also be changed to *Blocked* from *Running* as shown in Figure 5.

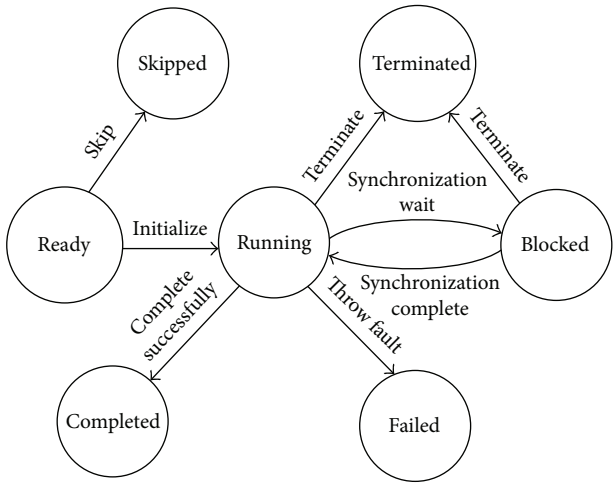


Figure 5: The state transition model of <*flow*> activity.

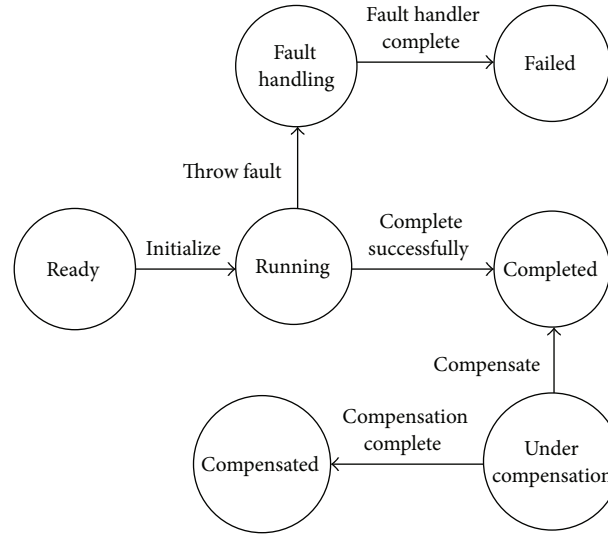


Figure 6: The state transition model of <*scope*> activity.

The <*sequence*> activity contains one or more activities that are performed sequentially. Given a <*sequence*> activity x, its initial state is *Ready*, and if the first inner activity has finished initialization and its state is *Running*, the state of x is changed from *Ready* to *Running*. Only if the last inner activity has completed successfully; that is, its state is *Completed*, the state of x is changed from *Running* to *Completed*. If an inner activity in activity x throws fault, the state of x is also changed to *Failed*. If x receives termination event, it will also forward this event for its inner activities, when only one inner activity of x is terminated successfully, and the state of x is changed to *Terminated*. In the course of the execution of x, if an inner activity z in activity x has the synchronization dependence on activity y (y is not the inner activity of x), z needs to wait for the successful completion of y. Therefore, activity x can be blocked because of the execution of y. In consequence, the state of x is changed from *Running* to *Blocked*. The transition process of its *Skipped* state is similar to <*flow*> activity. So the sate transition model of <*sequence*> activity is the same as <*flow*> activity, but the operational semantics of its state transition is only different from <*flow*> activity.

The <*if*> activity provides conditional behavior. The activity consist of an ordered list of one or more conditional branches defined by the <*if*> and optional <*else if*> elements. So the sate transition model of <*if*> activity is the same as <*sequence*> activity, but only the operational semantics of its state transition is different from <*sequence*> activity. When it selects one branch to execute, its state is changed from *Ready* to *Running*, and the state of the inner activities in other branches will be set to *Skipped*. Only when the state of the inner activity in any one branch is *Blocked*, its state will also be changed to *Blocked*.

The <*while*> activity, <*repeatuntil*> activity, and serial <*foreach*> activity provide iterative behavior. In BPEL specification, they cannot have the synchronization dependencies on other activities, so there is no the transition from *Ready* to *Skipped* in their state transition models.

And when their state transition semantic from *Running* to *Completed* is different from <*sequence*> activity, such transition must be satisfied with the completion condition and the successful completion of only one inner activity.

Parallel <*foreach*> activity will dynamically create and execute N+1 instances of the <*foreach*>'s enclosed <*scope*> activity as children in parallel until completion condition is satisfied. When completion condition is satisfied, if there are still the executing instances in the context, these instances will be forced to terminate. Because parallel <*foreach*> activity executes in parallel, its state transition model is the same as <*flow*> activity, but only its state transition semantic from *Running* to *Completed* is different from <*if*> activity, its transition must be satisfied with the completion condition or all instances' successful completion.

4.3. State Transition Model of Scope Activity. The <*scope*> activity provides the context of execution for its inner activities. Its difference from other structural activities is its special state transition model. The initial state of <*scope*> activity is *Ready*. When the main activity of <*scope*> activity has finished initialization and its event handler is also activated, its state is changed to *Running* from *Ready*. If in the course of execution the inner activities throw the exception, the fault handler will be triggered, and the state of <*scope*> activity will be changed to fault handling. In the course of such process, compensation handler is prohibited. When the fault handler has completed, its state is changed from fault handling to *Failed*. If <*scope*> activity has completed successfully, it means that this <*scope*> activity may be allowed to execute the compensation handler, once the compensation handler is triggered, its state is changed to under compensation from *Completed*. Until the compensation handler has completed, its state is also changed to compensated. Figure 6 shows the sate transition model of <*scope*> activity.

5. Modeling the Context of BPEL with State Transition System

In the above section, we discuss the temporal behavior of state transition and the *transition conditions* in BPEL activity. In general, the discussed state transition models correspond to state transition system, and such system may describe the dynamic process of the execution of BPEL, and the emerging states in the context can be evolved into new states after executing an action. The state transition of whole system is related to the following factors. The first factor is action. Action decides the state change of system, and the state transitions vary from the actions. Generally speaking, the system is related to input action and output action. Secondly, restriction also plays an important role in the state transition. If and only if the restriction conditions are satisfied, action can trigger the state transition of system, for example, in term with <*flow*> activity, the restriction of its successful completion is that its all inbuilt activities have completed successfully. Finally, the current state of system decides on the direction of state transition. Therefore, in this section, we present a state transition system which is used for modeling the context of BPEL activity.

Definition 1. The execution context of BPEL activity is a state transition system, which is defined as in the following seven tuples:

$$Context = \left(\mathcal{S}, \mathcal{S}_0, A_i, A_o, \mathcal{R}, \delta, \theta\right), \tag{1}$$

where

(1) $\mathcal{S}$ represents the finite state set of activity. If the type of activity is basic activity, then $|\mathcal{S}| = 1$; if the type of activity is structural activity, then $|\mathcal{S}| > 1$,

(2) $\mathcal{S}_0$ is an initial set of system, $\mathcal{S}_0 \subseteq \mathcal{S}$,

(3) A_i is a finite set of input action,

(4) A_o is a finite set of output action,

(5) $\mathcal{R}$ is a finite restriction set of state transition,

(6) δ is a function of state transition: $\mathcal{S} \times (A_i \cup A_o) \times \mathcal{R} \rightarrow \mathcal{S}$,

(7) θ is a mapping function: $\mathcal{CWS} \rightarrow Context$, and it will add an execution context for every composite Web service and uses the prosperities of composite Web service.

According to the above definition, the state transition semantic discussed in Section 4 may be formally described. And this section also presents the approaches to construct such state transition system.

5.1. Constructing the State Transition System of Basic Activity. Figure 3 represents the state transition model of the most basic activities. Their state transition results from fault event, compensation event, termination event, message event, and alarm event in BPEL activity. Therefore, it is necessary to define the message and action related to these events.

Action may be classified to two types: input action and output action, which correspond to *send* and *receive* operations. In terms of state transition system discussed in this paper, we need to do a research on the relations of state transition in the same activity or different activities. In general, input and output action will send the notifications or command of state transition, so action in Definition 1 may be defined as follows:

$$\mathcal{A} = (\mathcal{M}, \mathcal{O}), \tag{2}$$

where

(1) $\mathcal{M}$ is a message or event, $\mathcal{M} = (\mathcal{E}, \mathcal{T}_{\mathcal{M}})$, $\mathcal{T}_{\mathcal{M}} = \{Notification, Command\}$. $\mathcal{E}$ is command or notification. If $\mathcal{E}$ is command, it will send events such as *message*, *alarm*, *fault*, *compensation*, or *termination*. If $\mathcal{E}$ is notification, it will send states such as *Skipped*, *Ready*, *Running*, *Blocked*, *Failed*, *Terminated*, or *Completed*.

(2) $\mathcal{O}$ is a composite Web service, namely, activity. $\mathcal{O}$ is the source object of the sent message for input action, while it is the target object of the sent message for output action.

According to the above definition, we are able to formally describe the construction process of state transition system of basic activity. Let $S^t(x)$ be a predication, which indicates that activity x is in state t as

$$\begin{aligned} t \in \{&Ready, Running, Blocked, Completed,\\ &Failed, Terminated, Skipped\}\\ \cup \{&Faulthandling, Undercompensation, Compensated\} \end{aligned} \tag{3}$$

(A) $Ready \rightarrow Running$ $(x \in \mathcal{A}^{basic})$

(a) $R(x)$: $S^{Ready}(x)$,

(b) $A_i(x)$: *None*,

(c) $A_o(x)$: $(\mathcal{M}, z)$, $\mathcal{M} = (Running, Notification)$, $z = parents(x)$,

where

(i) $\mathcal{T}_{\mathcal{A}} = \{sequence, flow, pick, if, while, repeatuntil, foreach, scope, invoke, receive, reply, wait, assign, empty, throw, compensate, exit\}$.

(ii) $\forall t \in \mathcal{T}_{\mathcal{A}}$, $\mathcal{A}_t = \{a \in \mathcal{A} \mid Type\ (a) = t\}$ is a set of all activities of type t.

(iii) $\mathcal{A}^{basic} = \mathcal{A}_{receive} \cup \mathcal{A}_{reply} \cup \mathcal{A}_{invoke} \cup \mathcal{A}_{wait} \cup \mathcal{A}_{empty} \cup \mathcal{A}_{throw} \cup \mathcal{A}_{assign} \cup \mathcal{A}_{compemsation} \cup \mathcal{A}_{exit}$ is a set of basic activities.

(iv) $parents(x)$ is a set of immediate ancestor of x.

(B) *Running* $\rightarrow$ *Completed* ($x \in \mathcal{A}^{basic} \setminus (\mathcal{A}_{throw} \cup \mathcal{A}_{exit})$)

(a) $R(x)$: $S^{Running}(x)$,
(b) $A_i(x)$: *None*,
(c) $A_o(x)$: $(\mathcal{M}, z)$, $\mathcal{M}$ = (*Completed*, *Notification*), z = *parents*(x).

(C) *Running* $\rightarrow$ *Terminated*

(a) <*exit*> activity ($x \in \mathcal{A}_{empty}$)

(i) $R(x)$: $S^{Running}(x)$,
(ii) $A_i(x)$: *None*,
(iii) $A_o(x)$: $(\mathcal{M}, z)$, $\mathcal{M}$ = (*Termination*, *Command*), y = *Process*.

(b) Other basic activities ($x \in \mathcal{A}^{basic} \setminus (\mathcal{A}_{throw} \cup \mathcal{A}_{exit} \cup \mathcal{A}_{empty} \cup \mathcal{A}_{wait})$)

(i) $R(x)$: $S^{Running}(x)$,
(ii) $A_i(x)$: $(\mathcal{M}, y)$, $\mathcal{M}$ = (*Termination*, *Command*), y = *parents*(x),
(iii) $A_o(x)$: $(\mathcal{M}, z)$, $\mathcal{M}$ = (*Terminated*, *Notification*), z = *parents*(x).

(D) *Running* $\rightarrow$ *Failed* ($x \in \mathcal{A}^{basic} \setminus (\mathcal{A}_{exit} \cup \mathcal{A}_{empty})$)

(a) <*throw*> activity ($x \in \mathcal{A}_{throw}$)

(i) $R(x)$: $S^{Running}(x)$,
(ii) $A_i(x)$: *None*,
(iii) $A_o(x)$: $(\mathcal{M}, z)$, $\mathcal{M}$ = (*Failed*, *Notification*), z = *parentscope*(x),
where
parentscope(x) represents its immediate father <*scope*> activity.

(b) Other basic activities ($x \in \mathcal{A}^{basic} \setminus (\mathcal{A}_{exit} \cup \mathcal{A}_{empty} \cup \mathcal{A}_{throw})$)

(i) $R(x)$: $S^{Running}(x)$,
(ii) $A_i(x)$: *None*,
(iii) $A_o(x)$: $(\mathcal{M}, z)$, $\mathcal{M}$ = (*Failed*, *Notification*), z = *parents*(x).

5.2. Constructing the State Transition System of Structural Activity. Section 4.2 represents the state transition model of structural activities. The state transition of structural activities results from the state of their inner activities, input action and output action. The state of inner activities restricts the state transition of structural activity and partly decides the input action of structural activity. In consequence, we will discuss the restriction condition of state transition of structural activity, and input actions result in state transition and output actions. Because <*scope*> activity is special structural activity, so we will discuss how to construct its state transition system in the next subsection.

(A) *Ready* $\rightarrow$ *Running* ($x \in \mathcal{A}^{structured}$)

(a) $R(x)$: $S^{Ready}(x) \wedge \exists y(y \in children(x) \wedge S^{Running}(y)) \Rightarrow S^{Running}(x)$,
(b) $A_i(x)$: $(\mathcal{M}, y)$, $\mathcal{M}$ = (*Running*, *Notification*), $y \in children(x)$,
(c) $A_o(x)$: $(\mathcal{M}, z)$, $\mathcal{M}$ = (*Running*, *Notification*), z = *parents*(x),
where
(i) $\mathcal{A}^{structured}$ = $\mathcal{A}_{squence} \cup \mathcal{A}_{flow} \cup \mathcal{A}_{if} \cup \mathcal{A}_{pick} \cup \mathcal{A}_{while} \cup \mathcal{A}_{scope} \cup \mathcal{A}_{repeatuntil} \cup \mathcal{A}_{foreach}$ is a set of structured activities,
(ii) $children(x)$ is a set of immediate descendant of x.

(B) *Running* $\rightarrow$ *Completed*

(a) <*sequence*> activity ($x \in \mathcal{A}_{squence}$)

(i) $R(x)$: $S^{Running}(x) \wedge y = tail(x) \wedge S^{Completed}(y) \Rightarrow S^{Completed}(x)$,
(ii) $A_i(x)$: $(\mathcal{M}, y)$, $\mathcal{M}$ = (*Completed*, *Notification*), $y \in children(x)$,
(iii) $A_o(x)$: $(\mathcal{M}, z)$, $\mathcal{M}$ = (*Completed*, *Notification*), z = *parents*(x).

(b) <*flow*> activity ($x \in \mathcal{A}_{flow}$)

(i) $R(x)$: $S^{Running}(x) \wedge \forall y(y \in children(x) \wedge S^{Completed}(y)) \Rightarrow S^{Completed}(x)$,
(ii) $A_i(x)$ and $A_o(x)$ is the same as <*sequence*> activity.

(c) Iterative activity ($x \in \mathcal{A}_{repeatuntil} \cup \mathcal{A}_{while} \cup \mathcal{A}_{foreach}$)

(i) *CompletionCond*(x) represents the completion condition of activity,
(ii) $R(x)$: $S^{Running}(x) \wedge \exists! y(y \in children(x) \wedge S^{Completed}(y)) \wedge CompletionCond(x) \Rightarrow S^{Completed}(x)$,
(iii) $A_i(x)$ and $A_o(x)$ is the same as <*sequence*> activity.
(iv) $R(x)$: $S^{Running}(x) \wedge \exists! y(y \in children(x) \wedge S^{Terminated}(y)) \Rightarrow S^{Terminated}(x)$,
(v) $A_i(x)$ and $A_o(x)$ is the same as <*sequence*> activity.

(d) <*if* > activity ($x \in \mathcal{A}_{if}$)

(i) $R(x)$: $S^{Running}(x) \wedge \exists y(y \in children(x) \wedge S^{Completed}(y)) \Rightarrow S^{Completed}(x)$,
(ii) $A_i(x)$ and $A_o(x)$ is the same as <*sequence*> activity.

(C) *Running* $\rightarrow$ *Terminated*

(a) <*flow*> activity ($x \in \mathcal{A}_{flow}$)

(i) $R(x)$: $S^{Running}(x) \wedge \forall y(y \in children(x) \wedge S^{Terminated}(y)) \Rightarrow S^{Terminated}(x)$,
(ii) $A_i(x)$: $(\mathcal{M}, y)$, $\mathcal{M}$ = (*Termination*, *Command*), y = *parents*(x), $(\mathcal{M}, y')$, $\mathcal{M}$ = (*Terminated*, *Notification*), y' = *children* (x),

(iii) $A_o(x)$: $(\mathcal{M}, \mathcal{Z})$, $\mathcal{M}$ = (*Termination*, *Command*),

where
$\forall z \in \mathcal{Z}$, $S^{Running}(z) \wedge \mathcal{Z} \subseteq children(x)$.
$(\mathcal{M}, z')$, $\mathcal{M}$ = (*Terminated*, *Notification*), $z' = parents(x)$.

(b) <*sequence*> activity and <*if*> activity ($x \in \mathcal{A}_{squence} \cup \mathcal{A}_{if}$)

(i) $R(x)$: $S^{Running}(x) \wedge \exists y(y \in children(x) \wedge S^{Terminated}(y)) \Rightarrow S^{Terminated}(x)$,
(ii) $A_i(x)$: $(\mathcal{M}, y)$, $\mathcal{M}$ = (*Termination*, *Command*), $y = parents(x)$,
$(\mathcal{M}, y')$, $\mathcal{M}$ = (*Terminated*, *Notification*), $y' = children(x)$,
(iii) $A_o(x)$: $(\mathcal{M}, z)$, $\mathcal{M}$ = (*Termination*, *Command*), $z = children(x)$,
$(\mathcal{M}, z')$, $\mathcal{M}$ = (*Terminated*, *Notification*), $z' = parents(x)$.

(c) Iterative activity ($x \in \mathcal{A}_{repeatuntil} \cup \mathcal{A}_{while} \cup \mathcal{A}_{foreach}$)

(i) $R(x)$: $S^{Running}(x) \wedge \exists! y(y \in children(x) \wedge S^{Terminated}(y)) \Rightarrow S^{Terminated}(x)$,
(ii) $A_i(x)$ and $A_o(x)$ is the same as <*sequence*> activity.

(D) *Running* → *Failed*

(a) $R(x)$: $S^{Running}(x) \wedge \exists y(y \in children(x) \wedge S^{failed}(y)) \Rightarrow S^{failed}(x)$,
(b) $A_i(x)$: $(\mathcal{M}, y)$, $\mathcal{M}$ = (*Failed*, *Notification*), $y \in children(x)$,
(c) $A_o(x)$: $(\mathcal{M}, z)$, $\mathcal{M}$ = (*Failed*, *Notification*), $z = parents(x)$.

(E) *Running* → *Blocked*

(a) <*sequence*> activity ($x \in \mathcal{A}_{squence}$)

(i) $R(x)$: $S^{Running}(x) \wedge \exists! y \forall z(y \neq head(x) \wedge (z \in LR(z, l, y) \wedge (S^{Running}(z) \vee S^{Ready}(z)) \vee S^{Blocked}(y)) \Rightarrow S^{Blocked}(x)$,
(ii) $A_i(x)$: $(\mathcal{M}, y)$, $\mathcal{M}$ = (*Blocked*, *Notification*), $y \in children(x)$,
(iii) $A_o(x)$: $(\mathcal{M}, z)$, $\mathcal{M}$ = (*Blocked*, *Notification*), $z = parents(x)$.
(iv) $R(x)$: $S^{Blocked}(x) \wedge \exists! y \ \forall z(y \neq head(x) \wedge S^{Running}(x) \wedge (z \in LR(z, l, y) \wedge S^{Completed}(z)) \Rightarrow S^{Running}(x)$,
(v) $A_i(x)$: $(\mathcal{M}, y)$, $\mathcal{M}$ = (*Completed*, *Notification*), $y \in LR(y, l, x')$, $x' \in children(x)$,
(vi) $A_o(x)$: $(\mathcal{M}, z)$, $\mathcal{M}$ = (*Running*, *Notification*), $z = parents(x)$.

(b) Other structural activities ($x \in \mathcal{A}^{structured} \setminus \mathcal{A}_{squence}$)

(i) $R(x)$: $S^{Running}(x) \wedge \exists y(y \in children(x) \wedge S^{Blocked}(y)) \Rightarrow S^{Blocked}(x)$,
(ii) $A_i(x)$ and $A_o(x)$ is the same as <*sequence*> activity.

(F) *Blocked* → *Running*

(a) <*sequence*> activity ($x \in \mathcal{A}_{squence}$)

(i) $R(x)$: $S^{Blocked}(x) \wedge \exists! y \ \forall z(y \neq head(x) \wedge S^{Running}(x) \wedge (z \in LR(z, l, y) \wedge S^{Completed}(z)) \Rightarrow S^{Running}(x)$,
(ii) $A_i(x)$: $(\mathcal{M}, y)$, $\mathcal{M}$ = (*Completed*, *Notification*), $y \in LR(y, l, x')$, $x' \in children(x)$,
(iii) $A_o(x)$: $(\mathcal{M}, z)$, $\mathcal{M}$ = (*Running*, *Notification*), $z = parents(x)$.

(b) Other structural activities ($x \in \mathcal{A}^{structured} \setminus \mathcal{A}_{squence}$)

(i) $R(x)$: $S^{Blocked}(x)$,
(ii) $A_i(x)$: $(\mathcal{M}, y)$, $\mathcal{M}$ = (*Running*, *Notification*), $y \in children(x)$,
(iii) $A_o(x)$: $(\mathcal{M}, z)$, $\mathcal{M}$ = (*Running*, *Notification*), $z = parents(x)$.

5.3. Constructing the State Transition System of Scope Activity. Section 4.3 presents the state transition model of <*scope*> activity. The <*scope*> activity is equivalent to the scope of program language. It provides long-running transaction for process fragment in its *scope*. Because <*scope*> activity is also belonging to structural activity, its state depends on the states of its inner activities. According to Figure 6, the state transition system of <*scope*> activity is generated by the following approach.

(A) *Ready* → *Running* ($x \in \mathcal{A}_{scope}$)

(a) $R(x)$: $S^{Ready}(x) \wedge \exists y(y \in \mathcal{A}^{directenc}(x) \wedge S^{Running}(y)) \Rightarrow S^{Running}(x)$,
(b) $A_i(x)$: $(\mathcal{M}, y)$, $\mathcal{M}$ = (*Running*, *Notification*), $y \in \mathcal{A}^{directenc}(x)$,
(c) $A_o(x)$: $(\mathcal{M}, z)$, $\mathcal{M}$ = (*Running*, *Notification*), $z = parentscope(x)$,

where

$\mathcal{A}^{directenc}(x)$ is the set of all activities that are directly enclosed in *scope* x.

(B) *Running* → *Faulthandling* ($x \in \mathcal{A}_{scope}$)

(a) $R(x)$: $S^{Running}(x) \wedge \exists y(y \in \mathcal{A}^{directenc}(x) \wedge S^{Failed}(y)) \Rightarrow S^{Faulthandling}(x)$,
(b) $A_i(x)$: $(\mathcal{M}, y)$, $\mathcal{M}$ = (*Failed*, *Notification*), $y \in \mathcal{A}^{directenc}(x)$,
(c) $A_o(x)$: $(\mathcal{M}, z)$, $\mathcal{M}$ = (*Termination*, *Command*) $\forall z \in \mathcal{Z}$, $S^{Running}(z) \wedge \mathcal{Z} \subseteq \mathcal{A}^{directenc}(x)$.

(C) *Faulthandling* $\rightarrow$ *Failed* $(x \in \mathscr{A}_{scope})$

(a) $R(x)$: $S^{Faulthandling}(x) \wedge \forall y \forall z (y \in \mathscr{A}^{directenc}(x) \wedge (S^{Terminated}(y) \vee S^{Failed}(y) \vee S^{Completed}(y)) \wedge (z \in \mathscr{A}^{compensation}_{\mathscr{H}}(x) \wedge S^{Completed}(z))) \Rightarrow S^{Failed}(x)$,

(b) $A_i(x)$: $(\mathscr{M}, y)$, $\mathscr{M}$ = (*Terminated*, *Notification*), $y \in \mathscr{A}^{directenc}(x)$,

(c) $A_o(x)$: $(\mathscr{M}, z)$, $\mathscr{M}$ = (*Failed*, *Notification*), z = *parentscope*(x).

(D) *Running* $\rightarrow$ *Completed* $(x \in \mathscr{A}_{scope})$

(a) $R(x)$: $S^{Running}(x) \wedge \forall y (y \in (A^{mainset}(x) \cup \mathscr{A}^{fct}_{\mathscr{H}}) \wedge S^{Completed}(y)) \Rightarrow S^{Completed}(x)$,

(b) $A_i(x)$: $(\mathscr{M}, y)$, $\mathscr{M}$ = (*Completed*, *Notification*), $y \in \mathscr{A}^{directenc}(x) \cup \mathscr{A}^{fct}_{\mathscr{H}}$,

(c) $A_o(x)$: $(\mathscr{M}, z)$, $\mathscr{M}$ = (*Completed*, *Notification*), z = *parentscope*(x).

(E) *Completed* $\rightarrow$ *Under compensation* $(x \in \mathscr{A}_{scope})$

(a) $R(x)$: $S^{Completed}(x) \wedge \exists y (y \in \mathscr{A}^{compensation}_{\mathscr{H}} \wedge S^{Ready}(y)) \Rightarrow S^{Under\ compensation}(x)$,

(b) $A_i(x)$: $(\mathscr{M}, y)$, $\mathscr{M}$ = (*Compensation*, *Command*), y = *parentscope*(x),

(c) $A_o(x)$: $(\mathscr{M}, z)$, $A_o(x)$:$(\mathscr{M}, z)$, $\mathscr{M}$ = (*Compensation*, *Command*), $\forall z \in \mathscr{Z}$, $S^{Completed}(z) \wedge \mathscr{Z} \subseteq$ *childrenscope*(x),

where

childrenscope(x) are <*scope*> activities immediately enclosed by x.

(F) *Under compensation* $\rightarrow$ *Compensated* $(x \in \mathscr{A}_{scope})$

(a) $R(x)$: $S^{Undercompensation}(x) \wedge \forall y \forall z (y \in$ *childrenscope*$(x)) \wedge S^{Compensated}(y) \wedge (z \in \mathscr{A}^{compensation}_{\mathscr{H}}(x) \wedge S^{Completed}(z))) \Rightarrow S^{Compensated}(x)$,

(b) $A_i(x)$: $(\mathscr{M}, y)$, $\mathscr{M}$ = (*Compensated*, *Notification*), $y \in$ *childrenscope*(x) $(\mathscr{M}, y')$, $\mathscr{M}$ = (*Completed*, *Notification*), $y' \in \mathscr{A}^{compensation}_{\mathscr{H}}(x)$,

(c) $A_o(x)$: $(\mathscr{M}, z)$, $\mathscr{M}$ = (*Completed*, *Notification*), z = *parentscope*(x).

6. Conclusion and Future Work

In order to enhance the reliability of service composition applications in WSNs, we study the modeling approach of the execution context in the instance of BPEL for verifying the availability and operational semantics of exception handling and compensation mechanisms in BPEL instance for WSNs in this paper. At first, we discuss the possible states of a BPEL activity during the execution of the instance of BPEL. Secondly, we propose the respective state transition models for three types of BPEL activities and analyze the operational semantics for the state transitions. Finally, we present an approach to model state transition process by state transition system in order to describe and analyze the control flow resulted from exception handler and compensation mechanism for WSNs.

In the future, we plan to study how to evaluate the availability, operational semantics, and the efficiency of exception handling and compensation in BPEL instance for WSNs based on a rule system. So, a set of rules based on operational semantics for finding out the problems in exception handling and compensation are the key content in our future work. Simultaneously, we also plan to transform a state transition system of the instance of BPEL to a Petri net model. Thus, we can use Petri net theory and method to analyze and evaluate the availability, operational semantics, and efficiency of the state transition system easily.

Acknowledgments

This work was supported by the National Natural Science Funds Fund of China (61172084), the National High-tech R&D Program of China (2007AA01Z138), Natural Science Foundation of Hubei Province of China (2010CDB5201), Education Commission of Hubei Province, China (Q200625001), Xiangfan Municipal Science and Technique Foundation (2008GG1C41).

References

[1] E. Cerami, *Web Services Essentials: Distributed Applications with XML-RPC, SOAP, UDDI WSDL*, O'Reilly Associates, Sebastopol, Calif, USA, 2002.

[2] F. Curbera, W. A. Nagy, and S. Weerawarana, "Web services: why and how," in *Proceedings of the OOPSLA Workshop on Object-Oriented Web services*, pp. 34–40, ACM, 2001.

[3] A. Tsalgatidou and T. Pilioura, "An overview of standards and related technology in web services," *Distributed and Parallel Databases*, vol. 12, no. 2-3, pp. 135–162, 2002.

[4] S. Agarwal, S. Handschuh, and S. Staab, "Annotation, composition and invocation of semantic web services," *Web Semantics*, vol. 2, no. 1, pp. 31–48, 2004.

[5] WSDL, "Web Service Definition Language 1.1," http://www.w3.org/TR/wsdl.

[6] J. Koehler and B. Srivastava, "Web service composition: current solutions and open problems," in *Proceedings of Workshop on Planning for Web Services (ICAPS '03)*, pp. 28–35, ACM, 2002.

[7] M. Singh and M. Huhns, *Service-Oriented Computing: Semantics, Processes, Agents*, John Wiley & Sons, New York, NY, USA, 2005.

[8] R. Khalaf, N. Mukhi, and S. Weeravarana, "Service oriented composition in BPEL4WS," in *Proceedings of the International Conference on World Wide Web (WWW '03)*, ACM, 2003.

[9] M. Pistore, P. Traverso, P. Bertoli, and A. Marconi, "Automated synthesis of composite BPEL4WS web services," in *Proceedings of the IEEE International Conference on Web Services (ICWS '05)*, pp. 293–301, IEEE Computer Society, July 2005.

[10] K. H. Kim and C. A. Ellis, "Workflow performance and scalability analysis using the layered queuing modeling methodology," in *Proceedings of International ACM SIGGROUP Conference on Supporting Group Work*, pp. 135–143, ACM, October 2001.

[11] J. Jin and K. Nahrstedt, "On exploring performance optimizations in web service composition," in *Proceedings of the 5th ACM/IFIP/USENIX international Conference on Middleware*, vol. 78, pp. 115–134, Springer, 2004.

[12] D. Rud, A. Schmietendorf, and R. Dumke, "Performance modeling of WS-BPEL-based web service compositions," in *Proceedings of IEEE Services Computing Workshops (SCW '06)*, pp. 140–147, IEEE Computer Society, September 2006.

[13] S. Chandrasekaran, S. Ch, J. A. Miller, G. Silver, I. B. Arpinar, and A. Sheth, *Composition, Performance Analysis and Simulation of Web Services*, University of Georgia, Athens, Ga, USA, 2002.

[14] J. Q. Li, Y. S. Fan, and M. C. Zhou, "Performance modeling and analysis of workflow," *IEEE Transactions on Systems, Man, and Cybernetics Part A: Systems and Humans.*, vol. 34, no. 2, pp. 229–242, 2004.

[15] T. Liu, A. Behroozi, and S. Kumaran, "A performance model for a business process integration middleware," in *Proceedings of IEEE Conference on eCommerce*, pp. 191–198, IEEE Computer Society, 2003.

[16] A. K. Schomig and H. Rau, "A petri net approach for the performance analysis of business process," Tech. Rep. 116, Universität Würzburg, Schloss Dagstuhl, Germany, 1995.

[17] K. H. Kim and C. A. Ellis, "Performance analytic models and analyses for workflow architectures," *Information Systems Frontiers*, vol. 3, no. 3, pp. 339–355, 2001.

[18] Z. Wu, N. Xiong, J. H. Park, T. H. Kim, and L. Yuan, "A simulation model supporting time and non-time metrics for web service composition," *The Computer Journal*, vol. 53, no. 2, pp. 219–233, 2010.

[19] Z. Wu, Y. He, L. Zhao, and X. Peng, "A modeling method for web service composition on business layer," in *Proceedings of 4th International Conference on Networked Computing and Advanced Information Management (NCM '08)*, pp. 81–86, IEEE Computer Society, September 2008.

[20] M. Butler and C. Ferreira, "An operational semantics for StAC, a language for modeling long-running business transactions coordination models and languages," in *Proceedings of the 6th International Conference on Coordination Models and Languages (COORDINATION '04)*, vol. 2949 of *Lecture Notes in Computer Science*, pp. 87–104, Springer, Pisa, Italy, 2004.

[21] Y. He, L. Zhao, Z. Wu, and F. Li, "Modeling web services composition with transaction extension for performance evaluation," in *Proceedings of IEEE Asia-Pacific Services Computing Conference*, pp. 476–481, IEEE Computer Society, 2008.

[22] L. Bocchi, C. Laneve, and G. Zavattaro, "A calculus for long running transactions," in *FMOODS*, vol. 2884 of *Lecture Notes in Computer Science*, pp. 124–138, Springer, New York, NY, USA, 2003.

[23] M. Kovács, D. Varró, and L. Gönczy, "Formal modeling of BPEL workflows including fault and compensation handling," in *Proceedings of Workshop on Engineering Fault Tolerant Systems (EFTS '07)*, ACM, Dubrovnik, Croatia, September 2007.

[24] A. Charfi, B. Schmeling, and M. Mezini, "Transactional BPEL processes with AO4BPEL aspects," in *Proceedings of the 5th European Conference on Web services (ECOWS '07)*, pp. 149–158, IEEE Computer Society, November 2007.

[25] R. Lanotte, A. Maggiolo-Schettini, P. Milazzo, and A. Troina, "Design and verification of long-running transactions in a timed framework," *Science of Computer Programming*, vol. 73, no. 2-3, pp. 76–94, 2008.

[26] L. 5. Caires, C. Ferreira, and H. Vieira, "A process calculus analysis of compensations," in *Proceedings of the 4th international Symposium on Trustworthy Global Computing (TGC '08)*, C. Kaklamanis and F. Nielson, Eds., vol. 5474 of *Lecture Notes in Computer Science*, pp. 87–103, Springer, Barcelona, Spain, November 2008, Revised Selected Papers.

[27] B. Limthanmaphon and Y. Zhang, "Web service composition transaction management," in *Proceedings of the 15th Australasian Database Conference*, K. Schewe and H. Williams, Eds., vol. 52 of *ACM International Conference Proceeding Series*, pp. 171–179, Australian Computer Society, Dunedin, New Zealand, 2004.

[28] R. Bruni, H. Melgratti, and U. Montanari, "Theoretical foundations for compensations in flow composition languages," in *Proceedings of the 32nd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL '05)*, pp. 209–220, ACM, Long Beach, Calif, USA, January 2005.

[29] C. Eisentraut and D. Spieler, "Fault, compensation and termination in BPEL 2.0—a comparative analysis," in *Proceedings of the 5th International Workshop on Web Services and Formal Methods (WS-FM '08)*, R. Bruni and K. Wolf, Eds., vol. 5387 of *Lecture Notes in Computer Science*, pp. 107–126, Springer, Milan, Italy, September 2008, Revised Selected Papers.

[30] C. Ouyang, E. Verbeek, W. M. P. van der Aalst, S. Breutel, M. Dumas, and A. H. M. ter Hofstede, "Formal semantics and analysis of control flow in WS-BPEL," *Science of Computer Programming*, vol. 67, no. 2-3, pp. 162–198, 2007.

[31] R. Kazhamiakin, M. Pistore, and L. Santuari, "Analysis of communication models in web service compositions," in *Proceedings of the 15th International Conference on World Wide Web (WWW '06)*, pp. 267–276, ACM, Edinburgh, Scotland, May 2006.

[32] S. Nakajima, "Model-checking behavioral specification of BPEL applications," *Electronic Notes in Theoretical Computer Science*, vol. 151, no. 2, pp. 89–105, 2006.