A Distributed Task Allocation Strategy for Collaborative Applications in Cluster-Based Wireless Sensor Networks

Feng Wang,1,2,3 Guangjie Han,1,2,4 Jinfang Jiang,1 and Hao Qiu1

1 Department of Information & Communication Systems, Hohai University, Changzhou 213022, China
2 Guangdong Provincial Key Laboratory of Petrochemical Equipment Fault Diagnosis, Guangdong University of Petrochemical Technology, Maoming 525000, China
3 School of Information Science & Engineering, Changzhou University, Changzhou 213164, China
4 Changzhou Key Laboratory of Photovoltaic System Integration & Production Equipment Technology, Changzhou 213022, China

Correspondence should be addressed to Guangjie Han; hanguangjie@gmail.com

Received 25 March 2014; Accepted 19 May 2014; Published 8 June 2014

Academic Editor: Lei Shu

Copyright © 2014 Feng Wang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Balancing energy consumption and prolonging network lifetime have been hot topics in energy limited wireless sensor networks (WSNs). It is generally known that multiple sensor nodes collaborating with each other to perform tasks are an important means of energy balancing. In addition, a reasonable task allocation strategy is also an efficient method for task performing and energy saving. Furthermore, in order to save energy consumption, many WSNs are organized into clusters. Therefore, in this paper, we propose a distributed task allocation strategy for collaborative applications (DTAC) in cluster-based WSNs. In DTAC, the service abilities of sensor nodes are first evaluated and then the sensor nodes are classified into different ranking domains. Based on the service abilities of cluster members, the service ability of each cluster is calculated. A complex task can be first allocated to several clusters, and then scheduled on collaborative cluster members. Simulation results demonstrate that, compared with current related works, DTAC is much more efficient for complex task processing in terms of reducing energy consumption, shortening execution time, and balancing network loads.

1. Introduction

1.1. Background and Motivation. In recent years, WSNs have been widely used in many applications including military, industrial, household, medical, marine, and other fields, especially in natural disaster monitoring, early warning, rescuing, and other emergency situations [1]. With the development of WSNs, the related applications present higher real-time requirements. At the same time, the amounts of calculation and communication also become bigger, which introduces high energy consumption and short network lifetime. However, it is generally known that sensor nodes in WSNs are small with limited processing and computing resources [2]. It is impossible for a single sensor node to independently accomplish a complex task. Therefore, how to assign a complex task with a large amount of calculation to appropriate sensor nodes has become a pressing issue in energy limited WSNs.

There are two kinds of network structures for WSNs: flat and hierarchical networks. Compared with flat networks, hierarchical networks, for example, cluster-based WSNs, are much more energy efficient, where sensor nodes are classified into several clusters and managed by cluster heads. Therefore, we research on the task allocation topic in cluster-based WSNs.

Many task allocation methods have been proposed for WSNs [3–6]. Most of them only concentrate on reducing energy consumptions of sensor nodes. However, balancing all the node energy consumptions in network is considered to be a better choice for prolonging network lifetime rather than just reducing energy consumptions of several sensor nodes. In addition, a task allocation in WSNs must satisfy the following requirements.

(i) Real-time requirement: most applications of WSNs have real-time requirements. Thus task allocations
in these applications need to be completed in time. In order to ensure the real-time requirement of task allocation, the task processing speeds of sensor nodes should be considered.

(ii) Heterogeneous network requirement: heterogeneous network is a new trend in the development of WSNs, where sensor nodes have different initial energy levels and different abilities of computation, communication, processing, and storing. Therefore, in the process of task allocation, the heterogeneity of sensor nodes should be taken into account.

(iii) Flexibility requirement: due to different hardware configurations of sensor nodes and dynamic environment parameters, randomly deployed sensor nodes sense different data information. The sensory data is dynamically changed over time. Therefore, an efficient task allocation method should be dynamically adjusted according to different environment parameters.

(iv) Scalability requirement: WSNs are always deployed in unattended and unreliable environments, where sensor nodes are easily attacked and died out, which ultimately changes the network structure. Therefore, a task allocation method should be suitable for any kinds of WSNs with dynamic network structure.

Based on the above-mentioned requirements, it is observed that research on a realistic task allocation model for WSNs is still at its infancy. In this paper, we proposed a distributed task allocation strategy for collaborative applications in cluster-based WSNs. Through simulations, it is found that our task allocation strategy outperforms traditional related works in terms of reducing energy consumption, shortening execution time, and balancing network loads.

1.2. Paper Organization. The rest of the paper is organized as follows. Section 2 introduces related works about task allocation in WSNs. Section 3 gives system models including network model, task model, and energy consumption model. Section 4 presents overview of DTAC. Section 5 investigates details of DTAC. Section 6 presents simulation results. Finally, Section 7 concludes the paper.

2. Related Work

2.1. Existing Task Allocation Strategies. In recent years, numerous studies have been conducted for task allocation and scheduling. For example, in [7], Giannecchini et al. proposed an online task scheduling mechanism called CoRAI to allocate network resources to tasks of periodic applications in WSNs. Given the amount of available resources and the task set, CoRAI can determine the sampling frequency for all the tasks so that the frequencies of the tasks on each sensor are optimized subject to the previously evaluated upper-bound execution frequencies. Simulation results show that CoRAI can efficiently assign the available resources among all the active tasks. However, CoRAI does not address mapping tasks to sensor nodes. In addition, CoRAI fails to explicitly discuss energy consumptions of sensor nodes.

In [8], an energy-constrained task mapping and scheduling called EcoMapS is proposed, which incorporates channel modeling, concurrent task mapping, communication and computation scheduling, and sensor failure handling algorithm. First, a channel model is presented for single-hop WSNs. Then, based on this channel model, communication and computation are jointly scheduled in the initialization phase. Furthermore, a quick recovery algorithm is executed in the quick recovery phase in case of sensor node failures. EcoMapS efficiently minimizes the schedule length subject to the energy consumption constraint in single-hop clustered WSNs. However, EcoMapS does not provide the guarantee of execution deadline; thus it is not suitable for real-time applications.

Tian et al. developed an application-independent task mapping and scheduling solution in [9], which not only provides real-time guarantee, but also implements dynamic voltage scaling mechanism to further optimize energy consumption. Using a novel multihop channel model and a communication scheduling algorithm, computation tasks and associated communication events are scheduled simultaneously with a dynamic critical-path scheduling algorithm. The critical-path in a DAG is the one with the largest sum of computation and communication costs. However, the proposed solution is not suitable for heterogeneous WSNs, where constituent components are not known priori. Consequently, for each node and edge in the DAG, there is generally more than one possible cost for executing communication task, so that the critical-path may not be distinct.

In [10], a novel task allocation strategy called balanced energy-aware task allocation (BEATA) is proposed for collaborative applications running on heterogeneous networked embedded systems aiming at making the best trade-off between energy saving and schedule length. The trade-off is achieved by the defined energy-adaptive window (EAW), in which a sensor node with lower energy consumption and earlier finish time is chosen for task processing. Choosing processing nodes without considering residual energy brings about a consequence that some nodes are chosen frequently and died early. Therefore, BEATA cannot well balance energy consumptions of sensor nodes.

Another similar task allocation algorithm which concentrates on both energy saving and time constraint is proposed in [11]. The proposed algorithm is a two-phase task allocation technique based on queuing theory. In the first phase, tasks are equally assigned to sensor nodes to measure the service capabilities of them. In the second phase, tasks are specifically allocated to some sensor nodes according to their measured capabilities in such a way to reduce the total completion times of all the tasks in network. However, the residual energy of sensor nodes dynamically changes with the number of processing tasks, which results in dynamic capabilities of sensor nodes. Therefore, how to dynamically update capabilities of sensor nodes needs further research.

Allocating tasks to sensor nodes should consider not only reducing processing time and energy consumption, but also improving reliability of the entire task execution.
Therefore, a novel task allocation algorithm based on \(A^*\) algorithm is proposed to optimize reliability and cost of task execution [12]. In order to further save energy consumption, a greedy \(A^*\) algorithm is proposed based on the traditional \(A^*\) algorithm. Compared with centralized execution, both \(A^*\) and greedy \(A^*\) algorithms can distribute energy consumption more uniformly across sensor nodes. Relatively speaking, the greedy \(A^*\) algorithm is more scalable than \(A^*\) algorithm with increasing number of sensor nodes. However, the sensor nodes conducting task allocation consume much more energy than ordinary sensor nodes do. That is to say, the proposed algorithm does not work well in energy balancing.

In order to achieve a fair energy balance among sensor nodes and maximize the overall network lifetime, an auction-based strategy for distributed task allocation is proposed in [13]. The real-time distributed task allocation problem is formulated as an incomplete information reverse auction game. The task sensing and allocating node is called auctioneer. The task processing node is a bidder. Then, based on winner determination protocol (WDP), an energy and delay efficient distributed winner determination protocol (ED-WDP) is proposed to select winner nodes to perform tasks. To the best of our knowledge, this is the first work that considers a completely distributed framework for incentive compatible reverse auction-based task allocation with an effective distributed winner protocol for WSN applications.

In the above-mentioned related works, we can see that the usage of resources is usually highly related to the execution of tasks which consumes a certain amount of energy. Except reducing and balancing energy consumption or shorting task execution time, parallel processing among sensor nodes is considered to be a promising solution for efficient task allocation. In [14], a self-adapted task scheduling strategy is proposed to achieve parallel processing of tasks. First, a multiagent-based architecture for WSNs is proposed and a mathematical model of dynamic alliance is constructed for task allocation. Then, an effective discrete particle swarm optimization algorithm is proposed for the dynamic alliance, in which the task allocation is conducted based on the energy consumption of sensor nodes, the amount of network loads, and the execution time of tasks.

In [15], a dynamic task allocation is proposed for multihop WSNs with low node mobility. The proposed algorithm incorporates a fast task reallocation algorithm to quickly recover from network disruptions, such as node or link failures. In addition, an evolitional self-learning mechanism based on a genetic algorithm continuously adapts the system parameters in order to meet the desired application delay requirements, while also achieving a sufficiently long network lifetime. Since the algorithm runtime incurs considerable time delay while updating task assignments, an adaptive window size is introduced to limit the delay periods and ensure an up-to-date solution based on node mobility patterns. To the best of our knowledge, this is the first study for mobile multihop WSNs.

2.2. Comparing DTAC with Existing Task Allocation Strategies.
From the above-mentioned related works, we can find that existing task allocation strategies always only take one influencing factor, such as energy consumption or network load, into account for task allocation. However, other contributing factors, such as the residual energy level or the reliability of a sensor node, cannot be neglected. Therefore, a novel distributed and energy efficient task allocation strategy is proposed in this paper. Compared with the existing task allocation strategies, the contributions of DTAC are listed as follows.

(i) We layer the subtasks to different layers based on directed acyclic graph (DAG) task model. All the subtasks in the same layer can be processed at the same time, which can promise efficient task parallel processing to reduce execution time.

(ii) In each cluster, all cluster members are classified into different ranking domains based on their service abilities. The ranking domain of each sensor node can be dynamically adjusted to guarantee the success rate of task processing.

(iii) We study the evaluation of service ability on each cluster. Based on the service abilities of clusters, complex tasks can be allocated to different clusters to balance energy consumption and prolong network lifetime.

(iv) In the process of task allocation, an improved contract net consultation mechanism is proposed to narrow the scope of bidding, which can significantly reduce communication overheads.

3. System Models

3.1. Network Model. A cluster-based WSN is studied in this paper, where sensor nodes are classified into several clusters according to communication range. Each cluster is composed of one cluster head and several cluster members. Cluster heads have much more residual energy and longer communication range than cluster members. Each cluster member can only directly communicate with its cluster head, while one cluster head can not only directly communicate with its neighbor cluster heads, but also contact other cluster heads or the base station with multihop communication. The cluster-based WSN is considered to be heterogeneous, where sensor nodes have different initial energy levels and different processing speeds for tasks. However, the energy consumption of data transmission is assumed to be the same.

3.2. Task Model. A complicated task consists of several subtasks, which can be denoted by a directed acyclic graph \(G = (T, E, w, l)\) [3]. As shown in Figure 1, a DAG consists of a set of vertices \(T\) representing the tasks needs to be executed and a set of directed edges \(E\) representing dependencies among tasks. Each task has an execution deadline \(l_{t_j}\). In order to ensure the smooth completion of the complex task, each subtask must be completed before its execution deadline. If there is a directed edge \(e_{ij}\) from a vertex \(t_i\) to another one \(t_j\), the execution of subtask \(t_j\) needs the output result of the execution of subtask \(t_i\). Therefore, the subtask \(t_j\) needs to
be processed after the completion of $t_i$. The weight value $w_j$ of a vertex refers to the amount of computations of the task. The weight value $l_j$ of a directed edge refers to the communication traffic between two tasks. For the sake of simplicity, we assume that the amount of computations approximately equals the amount of transmitted data, which follows a normal distribution.

For an edge $e_{ij}$, $t_j$ is called an immediate predecessor of $t_i$. $t_i$ is called an immediate successor of $t_j$. A task without immediate predecessors is an entry task and a task without immediate successors is an exit task. If one task $t_j$ is scheduled on one node while its immediate predecessor $t_i$ is scheduled on another node, a communication between the two nodes is required. Therefore, $t_j$ cannot start its execution until the communication is completed and the result of $t_i$ is received. If both $t_i$ and $t_j$ are assigned on the same node, the communication latency is considered to be zero. $t_i$ can start execution after $t_j$ is finished. In addition, we assume that a subtask can only be assigned on one sensor node, but each sensor node can process several subtasks.

3.3. Energy Consumption Model. The energy consumption of a sensor node consists of two parts: energy consumption of task processing and communication overheads. When a task $t_j$ is processed by node $j$. The energy consumption of task processing is calculated by

$$C_{\text{comp}}(i, j) = P_c(j) \times t_i^j,$$

where $P_c(j)$ is the average energy consumption per second, $t_i^j$ is the processing time on node $j$, which is calculated by

$$t_i^j = \frac{w_i}{f_j},$$

where $w_i$ is the amount of computation for task $t_i$, $f_j$ is the task processing speed of node $j$.

The communication overheads also consist of two parts: energy consumption of transmitting and receiving data, the weight value $\omega_j$, which is calculated by

$$C_{\text{comm}}(i, j) = \frac{\xi_{\text{elec}} + \xi_{\text{amp}} \cdot d^2}{l},$$

where $\xi_{\text{elec}}$ and $\xi_{\text{amp}}$ are hardware-related parameters. $d$ is a defined threshold. When $d \geq d_0$, the energy consumption of transmitting greatly increases. Therefore, the communication overheads of node $j$ are given as $C_{\text{comm}}(i, j) = C_{\text{tx}}(l, d) + C_{\text{rx}}(l)$. Thus, the total energy consumption of node $j$ is obtained as follows:

$$C_{\text{total}}(V_j) = \sum_{i \in n} C_{\text{comp}}(i, j) + \sum_{i \in n} C_{\text{comm}}(i, j),$$

where $n$ is the number of processed tasks.

4. Overview of DTAC

4.1. Definitions

**Task Source Cluster.** A cluster sensing complex tasks is defined as a task source cluster. The cluster head in task source cluster is responsible for complex task allocation.

**Collaborative Cluster.** If a task source cluster does not have enough resources to complete a complex task, it turns to neighbor clusters asking for collaboration to complete the rest of subtasks. The neighbor cluster providing collaboration is called collaborative cluster.

The task allocation consists of two phases: (1) intercluster allocation and (2) intracluster allocation. A complex task can be divided into several subtasks. Intercluster allocation assigns each complex task in DAG to several collaborative clusters. Intracluster allocation schedules the subtasks on cluster members. In order to short task execution time, balance network loads, reduce energy consumption, and prolong network lifetime, the parallel processing of task is achieved by defining ranking domain.

4.2. Ranking Domain. In order to achieve the parallel-processing capability of a complex task, we propose a concept of ranking domain. A complex task can be divided into certain subtasks which can be assigned to different sensor nodes at the same time. Ranking domain is proposed to classify sensor nodes into different levels based on their service abilities. The sensor node with a higher service ability can be divided into a higher level ranking domain. Thus it can be selected with higher priority to perform processing tasks. The ranking domain classification is implemented based on sensor nodes’ reliability and ability. Here, node trust is proposed to represent the reliability of a sensor node and residual energy level denotes the ability. Next, the calculation of reliability and ability is introduced.

Trust management models have been recently suggested as an effective security mechanism for WSNs [18], which can
be used to evaluate the reliability of a sensor node. In this paper, node trust is defined as

$$D_C = \frac{N_{\text{success}}^i}{N_{\text{total}}^i},$$  (5)

where $N_{\text{success}}^i$ is the number of tasks successfully processed. $N_{\text{total}}^i$ is the total number of tasks cumulatively processed.

Residual energy is always considered to represent the ability of a sensor node. However, except the residual energy, the rate of energy consumption of a sensor node should also be taken into account. Therefore, the remaining energy ratio of a sensor node is proposed in this paper, which is the proportion of remaining energy accounting for the initial energy level. The remaining energy ratio of a sensor node is defined as

$$R_i = \frac{E_{\text{residual}}^i}{E_{\text{initial}}^i},$$  (6)

where $E_{\text{residual}}^i$ and $E_{\text{initial}}^i$ are the residual and initial energy of node $i$, respectively.

Hence, the node service ability can be calculated by

$$P_{Vi} = \alpha D_i + \beta R_i = \alpha \frac{N_{\text{success}}^i}{N_{\text{total}}^i} + \beta \frac{E_{\text{residual}}^i}{E_{\text{initial}}^i},$$  (7)

where $\alpha + \beta = 1$, $\alpha \in [0,1]$, $\beta \in [0,1]$, and $P_{Vi} \in [0,1]$.

Based on the service abilities, cluster members can be classified into three ranking domains $\{G_1, G_2, G_3\}$. First, the threshold $Th$ of remaining energy ratio is defined. If the remaining energy ratio of a sensor node is less than the threshold ($R_i < Th$), the sensor node does not have enough energy for task processing and thus is divided into the third ranking domain $G_3$. Otherwise, all the sensor nodes with enough service abilities are sorted from the highest value to the lowest value, and the selection rate $\alpha$ is defined, which is the percentage of sensor nodes in the first ranking domain. $\alpha$ can be dynamically adjusted according to the actual network parameters. Based on the sum number of the sensor nodes with enough service abilities and the selection rate $\alpha$, a part of sensor nodes can be selected as the first ranking domain nodes. The rest of sensor nodes are the second ranking domain nodes. However, if the number of the sensor nodes with enough service abilities is less than or equal to 2, all the sensor nodes are divided into the first ranking domain.

4.3. Service Ability Evaluation of Cluster. Based on the service abilities of cluster members, the service ability of a cluster $C_i$ can be calculated as follows.

First, the remaining energy ratio of a cluster is defined as

$$R_{C_i} = \frac{E_{\text{residual}}^{C_i} - C_{C_i}}{E_{\text{initial}}^{C_i}} \times 100\%,$$  (8)

where $E_{\text{residual}}^{C_i}$ is the current residual energy of the cluster. It is the sum of residual energy of all the cluster members, which can be calculated by

$$E_{\text{residual}}^{C_i} = \sum_{V_j \in C_i} E_{\text{residual}}^{V_j},$$  (9)

$C_{C_i}$ is the current energy consumption of the cluster, which can be calculated by

$$C_{C_i} = \sum_{V_j \in C_i} C_{V_j} / E_{\text{total}}^{C_i},$$  (10)

$E_{\text{initial}}^{C_i}$ is the initial energy level of the cluster, which can be calculated by

$$E_{\text{initial}}^{C_i} = \sum_{V_j \in C_i} E_{\text{initial}}^{V_j}.$$  (11)

Then, a threshold $Th_{C_i}$ of the remaining energy ratio is defined. If $R_{C_i} < Th_{C_i}$, the cluster can be a collaborative cluster. Otherwise, the cluster is considered to be inefficient for task processing.

Based on the remaining energy ratio of a cluster, the number of sensor nodes in the first and second ranking domains, and the service abilities of cluster members, the service ability of a cluster is defined as follows:

$$P_{C_i} = g_1 \times \sum_{V_j \in C_i} P_{V_j} \times \frac{R_{V_j} - Th}{Th},$$  (12)

where $g_1$ is the number of sensor nodes in the first ranking domain. We can conclude that the higher $P_{C_i}$ is the easier and $C_i$ can be selected as a collaborative cluster.

5. Details of DTAC

In this section, the details of DTAC are presented. DTAC consists of two steps. The first step is task layering. The second
step is task allocation. A complex task is first allocated to several clusters, which is called intercluster allocation. Then, in each cluster, subtasks are scheduled on different cluster members, which is called intracluster allocation.

5.1. Task Layering. As shown in Figure 2, based on DAG, a complex task can be layered to several layers as follows. (1) An entry task is assigned in the first layer. Its immediate successor task is assigned in the second layer. (2) For a subtask, if its immediate predecessor task is assigned in the ith layer, then the subtask can be assigned in the (i + 1)th layer. (3) For the tasks in the same layer, they are placed in an arrangement according to the order of their execution deadline \( t_{dl} \) from small to large. The task with smaller \( t_{dl} \) is placed in the front of the layer; therefore, it can be preferentially allocated to a sensor node with higher processing ability and completed within the deadline. After task layering, we can find that the tasks in the same layer do not have immediate predecessor or successor correlation. Therefore, they can be allocated to different nodes and concurrently processed, which ultimately reduces significant execution time.

5.2. Task Allocation. When a task source cluster senses a complex task, it first checks the number of sensor nodes in the first ranking domain. If the number of sensor nodes in the first ranking domain is greater than or equal to the number of subtasks, all the subtasks are allocated to the sensor nodes in the first ranking domain. Otherwise, the source cluster will ask neighbor clusters for collaboration based on their service abilities. If the collaborative clusters are selected,
the task allocation can be completed according to two steps: (1) intercluster allocation and (2) intraccluster allocation.

5.2.1. Task Intercluster Allocation. Task allocation begins from the first layer and then schedules the tasks in the next layer step by step. Only after the tasks in the upper layer are all scheduled, the tasks in the current layer can be allocated. We assume there are \( n \) subtasks in the \( i \)th layer, which are denoted by \( T_1, T_2, \ldots, T_n \). The number of sensor nodes in the first ranking domain of the source cluster is \( h \). The task allocation can be conducted as follows.

(i) If \( n \leq h \), all the tasks are allocated to the sensor nodes in the first ranking domain of the source cluster.

(ii) Otherwise, the first \( h \) tasks \( T_1, T_2, \ldots, T_h \) are allocated to the sensor nodes in the first ranking domain of the source cluster. Then, the source cluster asks for neighbor clusters’ collaboration by broadcasting collaboration request.

(iii) The neighbor clusters which receive the collaboration request first check their remaining energy ratio. For a neighbor cluster \( C_i \), if \( R_{C_i} \geq Th_C \), it replies the collaboration request and becomes a collaborative cluster. The reply message includes the service ability level \( P_{C_i} \) and the number of the sensor nodes \( g_i \) in the first ranking domain.

(iv) Once receiving the reply message, the source cluster places the collaborative clusters in an order according to their service abilities from small to large.

(v) The rest of tasks \( T_{h+1}, T_{h+2}, \ldots, T_n \) are allocated to the collaborative clusters according to the placed order. If

![Figure 4: Network equilibrium and the number of sensor nodes.](image-url)
the number of collaborative clusters is still not enough for task allocation, the source cluster will continually ask for neighbor clusters’ collaboration to process the rest of tasks until all the tasks are completed.

5.2.2. Task Intracluster Allocation. If a cluster receives several subtasks, the cluster head schedules the subtasks to its cluster members. In this paper, intracluster task allocation is implemented based on a contract net protocol [19]. In the negotiation process, the cluster head is a bidder which is responsible for task bidding. However, in conventional contract net protocol, the bidding is completed by broadcasting which wastes a large amount of energy. Therefore, in this paper, the ranking domain is proposed to reduce energy consumption and communication overheads by narrowing the scope of the bidding. That is, the cluster head first chooses the sensor nodes in the first ranking domain as objects of bidding. Only when the sensor nodes in the first ranking domain fail to complete the tasks, the sensor nodes in the second ranking domain are chosen as objects of bidding. Compared with the bidding with broadcast style, the improved bidding in this paper is much more energy efficient.

If a sensor node cannot reply the task processed results in time, the cluster head needs to replay bidding information to the sensor node. However, it is impossible for cluster head to repeat the bidding information without limit. Therefore, the maximum number of times of the replay is defined. If the number of relays gets to the defined maximum number, the cluster head will give up repeating the bidding regardless of whether the task is completed or not.

Figure 5: Task execution time and the number of sensor nodes.
6. Simulation

We use Matlab to evaluate the performance of DTAC. In our simulations, sensor nodes are randomly deployed in a square terrain (500 m × 500 m) with a base station. The base station is deployed at the center of the deployed area (250, 250) and assumed to have unlimited energy. Other simulation parameters are shown in Table 1.

We performed two different sets of simulations. In the first set experiment, we study two parameters’ impact on the performance of DTAC in terms of average residual energy, network equilibrium, and task execution time. The parameters are (1) the number of sensor nodes and (2) $\alpha$ which is the ratio of sensor nodes in the first ranking domain. In the second set experiment, we compare the performance of DTAC, EECNP [20], and ED-WDP [13], since the three algorithms are all proposed based on the contract net protocol [19].

6.1. Performance of DTAC. First, we simulate the performance of DTAC with different number of sensor nodes in terms of average residual energy, network equilibrium, and task execution time. Network equilibrium is used to evaluate...
the network load balancing, which is denoted by the variance of sensor nodes’ remaining energy.

6.1.1. Average Residual Energy and the Number of Sensor Nodes. We set the number of sensor nodes ranges from 125 to 1000. As shown in Figure 3, the residual energy decreases as the radio range increases. This is because more sensor nodes consume more resources, which leads to less residual energy. In addition, it is obvious that when the number of tasks ($N_{\text{task}}$) grows from 9 to 27, the residual energy decreases since processing more tasks needs higher energy consumption. Furthermore, with the increase of the number of simulation times, the residual energy of sensor nodes also decreases. However, the less the number of sensor nodes is, the faster the decrease of the residual energy is. Because, when the number of sensor nodes is small, they always need other nodes’ collaboration to process tasks. The collaboration process itself also consumes large amounts of energy, which thus greatly reduces the average remaining energy of the whole network. While more sensor nodes can provide more of total energy, thus in this case, the average residual energy decreases slightly.

6.1.2. Network Equilibrium and the Number of Sensor Nodes. As shown in Figure 4, we can see that more sensor nodes result in lower variance of sensor nodes’ remaining energy. It means that, with the same number of tasks, more sensor nodes can have better network load balancing. In addition,
the variance of remaining energy increases as the number of tasks increases. When the number of tasks increases, less sensor nodes result in higher variance of remaining energy. That is to say, more tasks need more sensor nodes to balance the network load. In order to balance the network load, we can dynamically adjust the density of sensor nodes according to the amount of tasks.

6.1.3. Task Execution Time and the Number of Sensor Nodes. As shown in Figure 5, we can conclude that the number of sensor nodes has little effect on the task execution time. In DTAC, all the tasks can be processed concurrently. It is unnecessary to deploy as many sensor nodes as possible for task processing. For example, in Figure 5, 125 sensor nodes are enough to complete tasks. In this case, more sensor nodes such as 500 or 1000 will not change the average task execution time. In addition, we can find the execution time decreases as the number of tasks increases. This is because the number of tasks in the same layer grows as the total number of tasks increases. The tasks in the same layer can be handled at the same time; thus the total execution time of tasks rapidly reduces. That is to say, DTAC is well suitable for processing many complex tasks.

6.1.4. Average Residual Energy and $\alpha$. In DTAC, the ranking domain is proposed for task parallel processing. The ranking domain is classified based on $\alpha$, which is the predefined ratio of sensor nodes in the first ranking domain. Therefore, in this
section, we simulate the performance of DTAC with different values of $\alpha$, which is set as $\alpha = 0.1$, $\alpha = 0.3$, $\alpha = 0.5$, $\alpha = 0.7$, and $\alpha = 0.9$.

Figure 6 shows that when the number of tasks is small ($N_{\text{task}} = 9$), the effect of $\alpha$ is not distinct, since the sensor nodes in the first ranking domain are enough for task processing. Even increasing $\alpha$ can provide more candidate sensor nodes, the number of actual used sensor nodes remains unchanged, and thus the average residual energy stays almost the same. However, we also find that, with the increasing number of tasks, greater $\alpha$ results in higher average residual energy. This is because when there are many tasks waiting for processing, increasing $\alpha$ provides more sensor nodes in the first ranking domain. The tasks can be completed in the process of intracluster allocation. Therefore, the energy consumption for intercluster allocation is saved.

6.1.5. Network Equilibrium and $\alpha$. Figure 7 shows that when the number of tasks is small ($N_{\text{task}} = 9$), the effect of $\alpha$ is not distinct. However, with the increasing number of tasks, greater $\alpha$ results in better network load balancing. Since greater $\alpha$ provides more sensor nodes for task allocation, the same amount of tasks can be scheduled on more sensor
nodes, which thus balances the energy consumption of sensor nodes.

6.1.6. Task Execution Time and $\alpha$. Figure 8 shows that when the number of tasks remains the same, the effect of $\alpha$ is not distinct. If the value of $\alpha$ is small, the bidding needs to be implemented among several collaborative clusters. Otherwise, the bidding only needs to be implemented within the source cluster. The difference is the execution time of intercluster and intracluster bidding. However, compared with the task processing time, the bidding time can be almost ignored. In addition, we can find that the task processing time is almost the small under different $\alpha$. That is to say, $\alpha$ greatly impacts the average energy consumption and network load balancing but has little influence on the task execution time. However, the task execution time decreases as the number of tasks increases. This is because the number of tasks in the same layer grows as the total number of tasks increases. The tasks in the same layer can be handled at the same time; thus the total execution time of tasks rapidly reduces.

6.2. Comparison of DTAC, EECNP, and ED-WDP. In this section, we compare the performance of DTAC with that of other two schemes: EECNP [20] and ED-WDP [13]. The number of sensor nodes is 250. $\alpha$ is set as 0.3.
6.2.1. Average Residual Energy. As shown in Figure 9, it is obvious that the performance of DTAC is better than that of the other two algorithms. Since DTAC can assign several tasks at the same time, EECNP and ED-WDP can only assign a task at a time. Therefore, for the same amount of tasks, DTAC is much more energy efficient. In addition, the energy efficiency of DTAC becomes more apparent with the increase of the number of simulation times. However, the energy consumption of DTAC is also increased as the number of tasks increases since much more energy is consumed for task bidding and communications.

6.2.2. Network Equilibrium. As shown in Figure 10, ED-WDP works worst. DTAC outperforms EECNP and ED-WDP. Since ED-WDP always chose the optimal sensor nodes to perform tasks, the frequently chosen nodes consume more energy than other nodes do; therefore the energy consumption of different sensor nodes is significantly different. While DTAC and EECNP are both taking the residual energy into account, thus they can work better in terms of balancing energy consumption. Compared with EECNP, DTAC can use more sensor nodes in neighbor collaborative clusters to allocate tasks. Therefore, DTAC works better than EECNP.

6.2.3. Task Execution Time. As shown in Figure 11, EECNP works worst. DTAC outperforms EECNP and ED-WDP. Since EECNP allocates tasks one by one, ED-WDP can allocate tasks while processing them. DTAC can provide task parallel processing. Therefore, DTAC is the best one with the shortest execution time.
7. Conclusion

In order to reduce average energy consumption of sensor nodes, balance network loads, shorten task execution time, and prolong network lifetime, this paper studied a complex task allocation problem and proposed a task allocation strategy called DTAC. DTAC consists of two steps: task layering and task allocation. Task allocation is implemented by intercluster allocation and intracluster allocation. In order to ensure parallel processing of task, the ranking domain is proposed based on sensor nodes’ service abilities to classify the sensor nodes into three levels. Sensor nodes with higher ranking domain can be preferentially selected to perform tasks. Based on the service abilities of cluster members, the service abilities of clusters can be calculated. Then, the task source cluster can choose several collaborative clusters to complete complex tasks according to the service abilities of them. Simulation results show that the proposed algorithm is much more energy efficient than related works. Also, it is quite suitable for complex task processing.

Conflict of Interests

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

Acknowledgments

The work is supported by “Natural Science Foundation of Jiangsu Province of China, no. BK20131137,” “Science & Technology Pillar Program (Social Development) of Changzhou Science and Technology Bureau, no. CE20135052,” and “Jiangsu Province Ordinary University Graduate Innovation Project, no. CXZZ13_02.”

References

[1] G. Han, H. Xu, T.Q. Duong, J. Jiang, and T. Haral, “Localization algorithms of wireless sensor networks: a survey,” Telecommunication Systems, vol. 52, no. 4, pp. 2419–2436, 2013.
[2] L. Shu, M. Hauswirth, L. Cheng, J. Ma, V. Reynolds, and L. Zhang, “Sharing worldwide sensor network,” in Proceedings of the 2nd International Workshop on SensorWebs, Databases and Mining in Networked Sensing Systems (SWDMNSS ’08) in Conjunction with the IEEE/IPSJ Symposium on Applications and the Internet (SAINT ’08), Turku, Finland, July-August 2008.
[3] Y. Tian, J. Boangoa, E. Ekici, and F. Ozguner, “Real-time task mapping and scheduling for collaborative in-network processing in DVS-enabled wireless sensor networks,” in Proceedings of the 20th International Conference on Parallel and Distributed Processing, pp. 1–10, April 2006.
[4] A. Y. Hamed, “Task allocation for maximizing reliability of distributed computing systems using genetic algorithms,” International Journal of Computer Networks and Wireless Communications, vol. 2, no. 5, pp. 560–569, 2012.
[5] H. Karimi, M. Kargahi, and N. Yazdani, “Energy-efficient cluster-based scheme for handling node failure in real-time sensor networks,” in Proceedings of the 8th IEEE International Symposium on Dependable, Autonomic and Secure Computing (DASC ’09), pp. 143–148, December 2009.
[6] M. Younis, K. Akkaya, and A. Kunjithapatham, “Optimization of task allocation in a cluster-based sensor network,” in Proceedings of the 8th IEEE International Symposium on Computers and Communications, vol. 1, pp. 329–334, June-July 2003.
[7] S. Gnanecchini, M. Caccamo, and C.-S. Shih, “Collaborative resource allocation in wireless sensor networks,” in Proceedings of the 16th Euromicro Conference on Real-Time Systems (ECRTS ‘04), pp. 35–44, July 2004.
[8] Y. Tian, E. Ekici, and F. Özgün, “Energy-constrained task mapping and scheduling in wireless sensor networks,” in Proceedings of the 2nd IEEE International Conference on Mobile Ad-Hoc and Sensor Systems (MASS ’05), pp. 211–218, November 2005.
[9] Y. Tian, Y. Gu, E. Ekici, and F. Özgün, “Dynamic critical-path task mapping and scheduling for collaborative in-network processing in multi-hop wireless sensor networks,” in Proceedings of the International Conference on Parallel Processing Workshops (ICPP’06), pp. 215–222, August 2006.
[10] T. Xie and X. Qin, “An energy-delay tunable task allocation strategy for collaborative applications in networked embedded systems,” IEEE Transactions on Computers, vol. 57, no. 3, pp. 329–343, 2008.
[11] M. Okhovvat, M. Sharifi, and H. Momeni, “Task allocation to actors in wireless sensor actor networks: an energy and time aware technique,” in Proceedings of the 1st World Conference on Information Technology (WCIT ’10), vol. 3, pp. 484–490, October 2010.
[12] B. S. Manoj, A. Sekhar, and C. S. R. Murthy, “A state-space search approach for optimizing reliability and cost of execution in distributed sensor networks,” Journal of Parallel and Distributed Computing, vol. 69, no. 1, pp. 12–19, 2009.
[13] N. Edalat, C.-K. Thom, and W. Xiao, “An auction-based strategy for distributed task allocation in wireless sensor networks,” Computer Communications, vol. 35, no. 8, pp. 916–928, 2012.
[14] W. Guo, N. Xiong, H.-C. Chao, S. Hussain, and G. Chen, “Design and analysis of self-adapted task scheduling strategies in wireless sensor networks,” Sensors, vol. 11, no. 7, pp. 6533–6554, 2011.
[15] Y. Jin, S. Vural, A. Gluhak, and K. Moessner, “Dynamic task allocation in multi-hop multimedia wireless sensor networks with low mobility,” Sensors, vol. 13, pp. 13998–14028, 2013.
[16] Y. Tian and E. Ekici, “Cross-layer collaborative In-network processing in multihop wireless sensor networks,” IEEE Transactions on Mobile Computing, vol. 6, no. 3, pp. 297–310, 2007.
[17] A. Wang and A. Chandrakasan, “Energy-efficient DSPs for wireless sensor networks,” IEEE Signal Processing Magazine, vol. 19, no. 4, pp. 68–78, 2002.

[18] G. Han, J. Jiang, L. Shu, J. Niu, and H.-C. Chao, “Management and applications of trust in wireless sensor networks: a survey,” Journal of Computer and System Sciences, vol. 80, no. 3, pp. 602–617, 2014.

[19] R. G. Smith, “The contract net protocol: high-level communication and control in a distributed problem solver,” IEEE Transactions on Computers, vol. 29, no. 12, pp. 1104–1113, 1980.

[20] Y. Li, Z. Gao, Y. Yang, Z. Guan, X. Chen, and X. Qiu, “A cluster-based negotiation model for task allocation in Wireless Sensor Network,” in Proceedings of the International Conference on Network and Service Management (CNSM ’10), pp. 112–117, October 2010.
