A PSO-optimized Real-time Fault-tolerant Task Allocation Algorithm in Wireless Sensor Networks

Wenzhong Guo, Jie Li, Senior Member, IEEE, Guolong Chen, Yuzhen Niu, and Chengyu Chen

Abstract—One of challenging issues for task allocation problem in wireless sensor networks (WSNs) is distributing sensing tasks rationally among sensor nodes to reduce overall power consumption and ensure these tasks finished before deadlines. In this paper, we propose a soft real-time fault-tolerant task allocation algorithm (FTAOA) for WSNs in using primary/backup (P/B) technique to support fault tolerance mechanism. In the proposed algorithm, the construction process of discrete particle swarm optimization (DPSO) is achieved through adopting a binary matrix encoding form, minimizing tasks execution time, saving node energy cost, balancing network load, and defining a fitness function for improving scheduling effectiveness and system reliability. Furthermore, FTAOA employs passive backup copies overlapping technology and is capable to determinate the mode of backup copies adaptively through scheduling primary copies as early as possible and backup copies as late as possible. To improve resource utilization, we allocate tasks to the nodes with high performance in terms of load, energy consumption, and failure ratio. Analysis and simulation results show the feasibility and effectiveness of FTAOA. FTAOA can strike a good balance between local solution and global exploration and achieve a satisfactory result within a short period of time.

Index Terms—wireless sensor networks, task allocation, fault tolerance, particle swarm optimization, primary/backup.

1 INTRODUCTION

Wireless sensor networks (WSNs) consist of a large number of sensor nodes, which usually collect important information in target environment. WSNs have been envisioned for a wide range of applications, such as battlefield intelligence, environmental tracking, and emergency response. In WSNs, one of the most important constraints on sensor nodes is low power consumption requirement. Sensor nodes have limited, generally irreplaceable, power sources [1]. Parallel processing among sensors is a promising solution to provide demanded computation capacity in WSNs. And task allocation and scheduling play an essential role in parallel processing. How to assign a task to the most appropriate sensor node and simultaneously balance the network load in context of the uncertain and dynamic network environments are important and urgent issues in WSNs.

As a typical problem of the area of high performance computing, task allocation and scheduling have been addressed in a variety of applications, such as multiprocessor system [2], grid computing [3], social networks [4], multihop wireless networks [5]. Although task allocation problem has been studied deeply in distributed systems, the problem for WSNs is different from traditional distributed systems. In WSNs, the challenge of task allocation is distributing sensing tasks rationally among sensor nodes to reduce overall power consumption while guaranteeing these tasks being finished before deadlines and prolonging network lifetime. Load balancing is a key point for prolonging the network lifetime. An inferior task allocation scheme will lead to overload of nodes and is harmful to the networks. Meanwhile, without proper task allocation strategy, each sensor node will just work individually, and all sensor nodes cannot work together in an energy efficient way. Due to challenging features and constraints of WSNs, such as environmental constraints, the dynamic topology, and the instability of wireless link, there exist more vulnerabilities and uncertainties for real-time applications in WSNs. Moreover, some sensor nodes may fail or be blocked. Because the failure of some sensor nodes should not affect the overall task of a sensor network, fault tolerance becomes a necessity for sustaining sensor networks functionalities without any interruption due to sensor node failures [1]. For example, if sensor nodes are being deployed in a battlefield for surveillance and detection, the fault tolerance has to be high because the sensed data are critical and sensor nodes can be destroyed by hostile actions. Therefore, providing a fault-tolerant mechanism for such application in WSNs is mandatory due to the critical nature of the tasks in the application [6].

Different layers of abstraction of WSNs have different fault-tolerant technologies. A summary of fault-tolerant technology in different layers of abstraction of WSNs is shown in Table 1. It can be seen that fault-tolerant
techniques of different layers have different functions. Application layer focuses on solving a series of applications depending on sensing tasks and reducing information redundancy with appropriate reliability. Most of existing work uses data aggregation to reduce data redundancy and acquire accurate data. However, data aggregation has some drawbacks such as unstable output value, conflicting with security and depending on network structure and aggregation model. Therefore, there is still a lot of work needed to be explored in application layer with innovation idea.

The primary/backup (P/B) copy technique is one of the most widely used fault-tolerant technologies. It allows copies of a task to run on different sensor nodes. Backup copy has two modes, namely, active mode and passive mode. Backup copy which can be executed simultaneously with primary copy is called active mode, or backup copy which is activated only after incorrect results are generated from the primary copy is called passive mode. In the P/B model, two copies of a task are executed serially on two different nodes and an acceptance test is performed to check the result [7]. Therefore, P/B is indeed a promising approach for fault tolerance in WSNs. Recently, there are many algorithms on task allocation for WSNs with the purpose of reducing task completion time and energy consumption. To the best of our knowledge, no work has been done on fault-tolerant task allocation for real-time task allocation in WSNs. In other words, it is a challenge to design a novel real-time fault-tolerant task allocation algorithm in WSNs. In our previous work [8], we proposed a task allocation algorithm for WSNs with particle swarm optimization (PSO) method and multi-agent technology, and P/B copy technique is preliminarily applied in proposed algorithm to provide fault tolerance mechanism. In this paper, we extend this work and further study potential issues in using P/B copy technique and backup copies overlapping technology to support fault tolerance of task allocation in WSNs. The major contributions of this study are summarized as follows:

- We develop a novel soft real-time fault-tolerant task allocation algorithm, FTAOA, to support WSNs.
- We employ P/B technology to tolerate permanent node failures and passive backup overlapping technology to reduce redundancy. Besides, two different utility functions are designed to measure comprehensive performance of sensor nodes for two copies of tasks.
- A discrete particle swarm optimization (DPSO) is designed in FTAOA to improve network reliability, reduce deadline missing ratio, minimize tasks execution time, save node energy cost, balance network load and prolong the lifetime of network.
- We conduct extensive experiments to compare the proposed FTAOA with the QoS-aware fault-tolerant (QAFT) scheduling algorithm [6], fault-tolerance task allocation algorithm without overlapping (FTAA), fault-tolerant task allocation with genetic algorithm (TAGA) and extended Min-Min (EMM) algorithm. In addition, by performing simulation experiments, we show the feasibility and effectiveness of FTAOA.

The rest of this paper is organized as follows. In Section 2, related work is provided. Section 3 describes the mathematical model of problem. In Section 4, proposed FTAOA algorithm is given in details. Section 5 analyzes the performance of the proposed FTAOA algorithm under a variety of scenarios and represents the simulation results. Finally, we make conclusions and discuss the future work in Section 6.

### 2 RELATED WORK

Task allocation is a critical issue for proper engineering of cooperative applications in wireless sensor networks (WSNs) with latency and energy constraints. Recently, there are several algorithms to solve task allocation problem of WSNs with the purpose of reducing task completion time and energy consumption. Jin et al. [5] proposed an adaptive intelligent task mapping together with a scheduling scheme based on a genetic algorithm (ITAS). They employed a hybrid fitness function in the algorithm to extend the overall network lifetime via workload balancing among collaborative nodes. Zeng et al. [9] developed an energy balanced directed acyclic graph (DAG) task scheduling algorithm and gave a genetic algorithm (GA) integrating chromosome coding to find approximate optimal solution. Both [5] and [9] adopt GA, but GA may easily get stuck in local optimum. Tian et al. [10] proposed localized cross-layer real-time task mapping and scheduling solutions for a dynamic voltage scaling (DVS) enabled WSN and modeled the wireless channel as a virtual node to execute communication tasks. Similarly, Yu et al. [11] developed an integer linear programming formulation and a polynomial time 3-phase heuristic in a single-hop cluster of homogeneous WSN by considering the time and energy costs and using DVS. However, their system model is quite different from ours. Based on Yu et al. [11], Tian et al. [12] developed a dynamic critical-path task mapping and

<table>
<thead>
<tr>
<th>Abstraction Structure</th>
<th>Objective</th>
<th>Techniques</th>
</tr>
</thead>
<tbody>
<tr>
<td>application layer</td>
<td>reducing the information redundancy with appropriate reliability</td>
<td>data aggregation, primary/backup copy</td>
</tr>
<tr>
<td>transport layer</td>
<td>finding out faulty nodes and improving monitoring quality</td>
<td>fault detection and isolation</td>
</tr>
<tr>
<td>network layer</td>
<td>providing better resilience and controlling traffic congestion</td>
<td>fault-tolerant routing</td>
</tr>
<tr>
<td>link layer</td>
<td>improving the coverage degree with robust link</td>
<td>fault tolerant coverage and topology control</td>
</tr>
<tr>
<td>physical layer</td>
<td>building reliability by exploring the natural information redundancy</td>
<td>hardware redundancy, multiple sensors exploration</td>
</tr>
</tbody>
</table>
scheduling (DCTMP) mode to guarantee application deadlines with minimum energy consumption. An optimal task scheduling algorithm (OTSAS-WSN) in a clustered wireless sensor network is proposed based on divisible load theory in [13]. OTSAS-WSN consists of two phases: intra-cluster task scheduling and inter-cluster task scheduling. The goals of [12] and [13] only focus on minimizing execution time and energy consumption. Xie et al. [14] proposed a novel task allocation strategy named balanced energy-aware task allocation (BEATA) for collaborative applications running on heterogeneous networked embedded systems. BEATA chooses a sensor node with lower energy consumption and earlier finish time for task processing. Choosing processing nodes without considering residual energy brings about a consequence that some nodes are chosen frequently and died early. Thus, BEATA cannot well balance energy consumptions of sensor nodes. In the contrary, our work defines an utility function to choose proper nodes and can well balance energy consumption. In order to prolong the lifetime of the network, Abdelhak et al. [15] proposed a balanced energy aware task allocation algorithm in WSNs. But in order to adapt to the dynamic network, it has to frequently collect updates from nodes, which incurs large overhead. Habib et al. [16] proposed a novel graph-based model to aggregate data from sensors to gateways. By using reverse auction-based task allocation mechanism, Neda et al. [17] proposed an energy and delay efficient winner determination protocol (ED-WDP), and designed an energy and delay efficient decision making method in front of the two-phase ED-WDP. But, the strategy they adopted is quite different from ours. Morteza et al. [18] took both energy awareness and reduction of actor tasks’ time to completion in wireless sensor and actor networks into account and proposed a two-phase task allocation technique based on queuing theory. However, the limited size of the queue associated with each actor in the real world was ignored. Yang et al. [19] proposed a modified version of binary particle swarm optimization (MBPSO) algorithm. They took task workload and connectivity as constraints for the problem. But they didn’t consider the fault-tolerant mechanism. An online task scheduling mechanism named CoRAI was proposed in [20] to allocate network resources to tasks of periodic applications in WSNs. CoRAI can efficiently assign available resources among all active tasks. But, Mapping tasks to sensor nodes was not addressed. Besides, [20] fails to explicitly discuss energy consumptions of sensor nodes. Tian et al. [21] presented an algorithm named EcoMapS which was a energy-constrained task mapping and scheduling solution for one-hop clustered homogeneous wireless sensor networks. EcoMapS efficiently minimizes the schedule length under energy consumption constraint. However, EcoMapS does not ensure to meet the task execution deadline; therefore, it is not suitable for real-time applications. Allocating tasks to sensor nodes should not only take minimizing processing time and energy consumption into account, but also improve reliability of entire networks. However, those aforementioned algorithms do not consider fault tolerance. Efficient fault-tolerant task allocation algorithms are capable of improving the schedulability, reliability, and flexibility of system [6]. Therefore, in this study, we focus on fault-tolerant task allocation algorithm of WSNs.

A wide variety of scheduling algorithms have been developed to provide fault tolerance for real-time systems in the past decade. P/B copy technology is a common methodology used for those fault-tolerant scheduling algorithms. Zheng et al. [22] proposed two strategies to improve scheduling and overloading efficiency and developed two algorithms to schedule backups of independent jobs and dependent jobs, respectively. [23] proposed a performance-driven load balancing algorithm with P/B approach for computational grids with low communication cost and replication cost. Chen et al. [24] proposed a novel fault-tolerant rate-monotonic best-fit (NRMFB) algorithm based on distributed control systems. The NRMFB algorithm employed a back tracing strategy to reassign the primary copy when no existing processors can accommodate the current backup copy. Zhu et al. [6] developed a novel QoS-aware fault-tolerant model and designed a new dynamic real-time QoS-aware fault-tolerant scheduling algorithm to support heterogeneous clusters. Luo et al. [25] proposed Teros and Debus techniques. When primary copies are successfully finished, Teros is responsible for terminating the execution of active-backup corresponding copies and Debus is responsible for scheduling active-backup copies as late as possible to defer the execution of backup copies.

### 3 Problem Formulation

The notations used in this paper are listed in Table 2.

In this paper, a task is an entity which fulfills a given specific function and is a basic unit of task allocation. In order to make full use of system resources, WSNs usually decompose an application into several tasks and then allocate them to multiple nodes to be executed. In this paper, we consider a static single-hop cluster of WSNs in a two-dimensional field. Sensor nodes are randomly distributed in the sensor field. The goal of task allocation can be simply described as follows. Generally, let \( m \) tasks be allocated reasonably to \( n \) sensors in order to minimize task execution time, save the energy consumption of nodes, balance the network load, prolong the network lifetime,
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2014.2386343, IEEE Transactions on Parallel and Distributed Systems

Given a task

time and deadline time of task $t$ for fault-tolerant task $T$ of nodes in a wireless sensor network. A set of nodes. Without loss of generality, we assume that all nodes have precedent constraints are equivalent to independent tasks by modifying arriving times and deadlines of dependent tasks. And the communication cost that is omitted is integrated into computation cost of tasks.

At one time instance, only one node may fail before another node fails. Those primary copies of tasks on failed node will be successfully finished by their backup copies. The correction of this supposition lies in that the large set of nodes are divided into several small groups and each group can tolerate one node failed.

Formally, we consider a set $N = \{n_1, n_2, n_3, ..., n_n\}$ of nodes in a wireless sensor network. A set $T = \{t_1, t_2, ..., t_m\}$ is used to denote $m$ real-time independent tasks. Since P/B technique is used in our fault-tolerant task allocation mode, each task $t_i$ has two copies (primary copy $t_i^P$ and backup copy $t_i^B$) executed on two different nodes. Given a task $t_i \in T$, $a_i$ and $d_i$ are used to denote the arrival time and deadline time of task $t_i$, respectively. $l_i$ denotes the laxity of $t_i^P$, $l_i = d_i - f_i^P$.

To make the following discussion more concrete, an example is given in Fig.1. A circle represents one or more nodes. The value in the circle is the number of nodes. A rectangle represents a task queue of the corresponding nodes. Without loss of generality, we assume that all nodes work at the same time. According to the definition of active mode and passive mode, $t_i^B$ and $t_i^B$ adopt active mode. $t_i^B$ uses passive mode. If the backup copy is passive mode, it does not be executed usually. Then, the total execution time to finish all tasks is composed of execution time of the primary copies and execution time of the active backup copies for $m$ tasks, it can be defined as:

$$Time = \sum_{i=1}^{m} et_{i,p}(t_i^p) + \sum_{i=1}^{m} S(t_i^B) \times (f_i^P - lst_{i,p}(t_i^p))$$ (1)

where $S(t_i^B)$ represents the mode of $t_i^B$. If $l_i$ is not less than $et_{i,p}(t_i^p)$, $t_i^B$ need not to be executed before $f_i^P$ for fault-tolerant. Because even $n_{i,p}(t_i^p)$ fails before the time $f_i^P$, $t_i^B$ also can be finished before $d_i$. Thus, it uses passive mode. Otherwise, $t_i^B$ needs to be executed before $f_i^P$ for fault-tolerant. Thus it employs active mode. Let $S(t_i^B)$ is equal to 1, which means $t_i^B$ uses active mode. Otherwise, $t_i^B$ uses passive mode. $S(t_i^B)$ can be expressed as:

$$S(t_i^B) = \begin{cases} 1, & \text{if } (l_i < et_{i,p}(t_i^p)) \\ 0, & \text{else} \end{cases} (1 \leq i \leq m)$$ (2)

Generally, when the primary copy is executed successfully, the corresponding backup copy need not to be executed any more. Thus the energy consumption of task execution contains the energy of the primary copy and the part the active backup copy consumed for $m$ tasks, and it can be expressed as:

$$Energy = \sum_{i=1}^{m} ene_{i,p}(t_i^p) + \sum_{i=1}^{m} S(t_i^B) \times ene_{i,p}(t_i^p) \times per_{i,p}(t_i^p)$$ (3)

where $per_{ij} = \frac{t_i^B - lst_{i,j}}{et_{i,j}} \times 100\%$ denotes the percentage of execution time that active mode $t_i^B$ executed on node $n_{ij}$ to the total time that the task executed on node $n_{ij}$.

To prolong the network lifetime, load balance should be considered in a good task allocation algorithm in WSNs. Here, the degree of load balance can be represented as:

$$Balance = \sum_{i=1}^{n} \frac{|b_i - \bar{b}|}{b_i}$$ (4)

where $b_i$, the load of $n_i$, denotes the time that $n_i$ successfully executes all tasks in its task queue. And $\bar{b}$ denotes the average load of nodes in the network. As an example, if the execution time of task queue of each node in Fig.1 is equal to each other, it is the best case. The load of network is balanced.

According to [26], reliability is defined as the probability that none of real-time tasks fail, even in the presence of hardware failures. However, the execution mode of backup copy is not considered in [26]. Therefore, the modified reliability model is given as follows:

$$RC = 100 \times (RC(zp) + RC(zb))$$ (5)

$$RC(zp) = \sum_{i=1}^{m} \lambda_p(t_i^p) \times et_{i,p}(t_i^p)$$ (6)

$$RC(zb) = \sum_{i=1}^{m} S(t_i^B) \times \lambda_p(t_i^p) \times (f_i^P - lst_{i,p}(t_i^p))$$ (7)

$$Rel = exp(-RC)$$ (8)
where $Rcl$ denotes the reliability of the network, $RC$ denotes the total reliability cost of the network, $RC(zp)$ and $RC(zb)$ are the total reliability costs of primary copy and backup copy of $m$ tasks, and $\lambda_i$ denotes the failure probability of $n_i$. Take Fig. 1 as an example, if the failure probability of $n_2$ is relatively large and it also needs to work a long time, it is obvious that the reliability of the system must be declined. Thus, the reliability of networks decreases with the increase of the reliability cost as shown in Equation (8).

Finally, the failure ratio of task scheduling can be expressed by the ratio $FR$ of missing the deadline for both primary and backup copies, and it can be defined as:

$$ FR = \frac{2 \times m}{\sum_{i=1}^{m} (1 \{ p_{fail_i} = 1 \} + \sum_{i=1}^{m} 1 \{ b_{fail_i} = 1 \}) } $$

(9)

where $p_{fail_i} = 1$ denotes the primary copy $t^p_i$ missing the deadline of task $t_i$. Likewise, $b_{fail_i} = 1$ denotes $t^b_i$ missing the deadline of task $t_i$.

According to the above descriptions, task allocation is an optimization problem that can be formalized as follows.

$$ \text{Maximize}(lt) \text{ S.T.} \left\{ \sum_{i=1}^{n} y_i \leq num \times 20\% \right\} $$

(10)

where $lt$ denotes the lifetime of WSNs and $num$ denotes the total number of the sensor nodes. $y_i=1$ means that node $n_i$ is failure and $y_i=0$ means that node $n_i$ is normal. Assume that WSNs is paralysis when the number of failure nodes exceeds 20% of the total number. For example, if $n_1$, $n_4$ and $n_8$ break down in Fig.1, then $\sum_{i=1}^{n} y_i = 3$ is larger than $num \times 20\% = 2$, therefore the network is paralysis.

## 4 PROPOSED ALGORITHM

Particle swarm optimization (PSO) is a relatively recent heuristic optimization technique developed by Kennedy and Eberhart [27]. The advantages of PSO over many other optimization algorithms are its ease of implementation and ability to converge to a reasonably good solution quickly. PSO is also more efficient in preserving population diversity to avoid premature convergence issue. GA is also an optimization approach which has been widely applied in various fields such as workflow applications scheduling [28]. A great number of experimental results show that PSO can solve nearly all kinds of optimization problems that can be solved by GA and other optimization approaches, thus it is indeed a powerful and vital optimization tool.

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. Kulakarni et al. [29] introduced PSO and discussed its suitability for WSN applications. They also presented a brief survey of how PSO is tailored to address these issues. In our previous work [8], [30] and [31], we used PSO to solve the topics we are interested in and achieved better results relative to the existing methods. In this paper, PSO method is also employed, and then we propose a FTAOA algorithm in a wireless sensor network environment. The flow chart of the algorithm is shown in Fig.2. It can be seen from Fig.2 that we can find that sink node collects tasks firstly, then it generates position and velocity in parallel for one generation of DPSO. Next, it begins a series of operations. After that if the terminal condition is not met, the next iteration begins. Otherwise, it publishes tasks and waits. The right part of Fig.2 is the execution process of tasks. If the primary version is finished successfully, no matter the mode of corresponding backup version is active or not, it does not have to be executed. If the mode of backup copy is passive, it is executed only when the primary copy fails. A simple example to show the proposed algorithm can be seen in Section 4.5.

To facilitate the presentation of FTAOA algorithm, some theorems and properties are introduced. Let $TimeSlot(t^b_i)$ and $TimeSlot(t^p_i)$ represent the executive time slot of $t^b_i$ and $t^p_i$, respectively.

**Property 1:** P/B technology is used in proposed FTAOA algorithm to achieve fault-tolerant mechanism by doing a redundant backup for each task and allocating the two copies of a task to different nodes at the same time.

**Proof:** Suppose that fault tolerance can be achieved if the primary copy $t^p_i$ and backup copy $t^b_i$ of task $t_i$ have been allocated to the same node. When $n_{p(t^p_i)}$ fails, the backup copy $n_{p(t^b_i)}$ of $t_i$ needs to be executed. However, $n_{p(t^b_i)}$ also fails for $n_{p(t^p_i)}$, thus failing to realize fault tolerance.

**Theorem 1:** $\forall n_k (n_k \in N)$, if $\exists t^p_i (t_j \in T)$, needed to be added into the task queue of $n_k$, fault tolerance can be realized if and only if $t^b_i$ is not overlapped with any copy of other tasks in the task queue.

**Proof:** Suppose $\exists t^p_i (t_j \in T)$, needed to be added to the node $n_k$ task queue. As shown in Fig.3, there are three cases of overlap. If $TimeSlot(t^b_i) \cap TimeSlot(t^p_j) \neq 0$ (Fig.3(a)), it is obvious that both two primary copies should be executed. However, since only one copy can be executed at the same time slot, fault tolerance cannot be achieved. Likewise, if $TimeSlot(t^b_j) \cap TimeSlot(t^p_i) \neq 0$ and $S(t^b_j) = 1$, as shown in Fig.3(b), both $t^b_j$ and $t^p_i$ need to be executed again, so fault tolerance also cannot be achieved. Scenario 3 in Fig.3(c) shows that $TimeSlot(t^b_i) \cap TimeSlot(t^p_j) \neq 0$ and $S(t^b_j) = 1$. Here we assume that $t^b_j$ fails to be executed, thus $t^b_i$ should be executed while $t^p_j$ also needs to be performed, then fault tolerance fails.

**Theorem 2:** $\forall t_k, t_i (t_i, t_k \in T)$, if $n_{p(t^b_i)} = n_{p(t^p_j)} = n_i$ and $n_{p(t^b_i)} = n_{p(t^p_j)} = n_j$, (ni $\neq$ nj), then fault tolerance cannot be guaranteed to realize if passive modes of these two backup copies $t^b_i$ and $t^b_j$ overlap.

**Proof:** Suppose that fault tolerance can be realized when $TimeSlot(t^b_i) \cap TimeSlot(t^b_j) \neq 0$. If $n_i$ fails, both two backup copies $t^b_i$ and $t^b_j$ allocated to $n_i$ must be executed. However, they are overlapping, and cannot be executed simultaneously at the same slot on a same node. Therefore, the assumption is incorrect. Thus Theorem 2 is
Lemma 1: Fault tolerance can be realized if and only if the overlapping of backup copy only occurs among passive backup copies.

\[ n_k \]

\[ (a) \text{ Scenario 1} \quad (b) \text{ Scenario 2} \quad (c) \text{ Scenario 3} \]

**Proof:** Suppose that \( t_i^B \) and \( t_i^B \) are backup copies allocated on the same node. According to Theorem 3, we have \( n_p(t_i^p) \neq n_p(t_i^p) \). First, if \( t_i^B \) and \( t_i^B \) are both active backup copies, and can overlap with each other, then the two copies should be executed in the same period according to the property of active backup copy. Since only one copy is allowed to be executed actually on one node at a time, the assumption is incorrect. Second, if only one of the two overlapping copies \( t_k^B \) and \( t_i^B \) is active, for example \( t_k^B \), and node \( n_p(t_i^p) \) which is responsible for \( t_i^p \) fails, then the passive backup copy \( t_k^B \) of \( t_i \) must be executed to achieve fault tolerance. As active backup copy of task \( t_k \), \( t_k^B \) must also be executed to gain fault tolerance. However, \( t_k^B \) overlaps with \( t_i^B \) and only one copy is allowed to be executed on the same node in the same period, so the assumption is incorrect.

**4.1 Task Allocation Mechanism Based on DPSO**

PSO was originally applied in continuous space optimization problem. However, task allocation is a discrete optimization problem, a DPSO model should be constructed by expanding the basic PSO in a binary space. Inspired
by the idea of PSO method, we can find that a task allocation problem can be described as a binary encoding in a matrix and the corresponding fitness function can be defined with task execution time, energy consumption, balance of network energy distribution and reliability cost as optimization objects to guide the evolution for optimal solution.

4.1.1 The Encoding of Particle

The position and velocity of particle are represented as two \( m \times n \) dimensional matrixes \( X \) and \( V \), respectively, and the particle position encoding is shown as follows:

\[
    x_{ij} = \begin{cases} 
        1, & \text{if } j^{th} \text{ node take part in } i^{th} \text{ task} \\
        0, & \text{else}
    \end{cases} \quad \text{(11)}
\]

where \( 1 \leq i \leq m; 1 \leq j \leq n \).

At each step, particles are manipulated according to the following formulas:

\[
    V^{t+1}(i) = w V^t(i) + c_1 r_1 (pBest(i) - X^t(i)) + c_2 r_2 (gBest - X^t(i)) \quad \text{(12)}
\]

\[
    X^{t+1}(i) = \begin{cases} 
        1, & \text{if } \text{rand}(i) < \text{sigmoid}(V^{t+1}(i)) \\
        0, & \text{else}
    \end{cases} \quad \text{(13)}
\]

where \( t \) is the current iteration times, \( X(i) \) and \( V(i) \) denote the position and velocity of the \( i \)-th particle, \( pBest(i) \) is the best position of particle \( i \), \( gBest \) is the global best position, \( w \) is inertia weight, \( c_1 \) and \( c_2 \) denote acceleration factors, \( r_1 \) and \( r_2 \) are two random numbers in the range of [0,1], \( \text{rand}(i) \) is the random function for generating random number in the range of [0,1], and \( \text{sigmoid}(V) = 1/(1 + \exp(-V)) \).

As an important parameter, an appropriate inertia weight \( w \) can help to obtain the balance between global and local search. In order to improve the global search performance of PSO, \( w \) updates according to the classical linear descending method [32]:

\[
    w = w_{\text{max}} - c_{\text{iter}} \times \frac{w_{\text{max}} - w_{\text{min}}}{m_{\text{iter}}} \quad \text{(14)}
\]

where \( c_{\text{iter}} \) denotes the current iteration times, \( m_{\text{iter}} \) denotes the maximum iteration times, \( w_{\text{max}} \) and \( w_{\text{min}} \) are the initial and terminal inertia weights, respectively.

4.1.2 Fitness Function

In section 3, we have described that task allocation in wireless sensor network is a multi-objective optimization problem (MOP). Here we transform the problem into a single-objective optimization problem by means of weighted sum strategy. The fitness function is shown as (15) and \( \alpha, \beta, \gamma \) and \( \theta \) are weighting factors.

\[
    \text{Fitness} = \alpha \times \text{Time} \times (1 + \text{FR}) + \beta \times \text{Energy} + \gamma \times \text{Balance} + \theta \times \text{RC} \quad \text{(15)}
\]

4.1.3 DPSO-Based Task Allocation Algorithm Description

Algorithm 1 shows the pseudo code of DPSO-based tasks allocation procedure. \( i \) refers to the serial number of current particle. \( j \) and \( m \) denotes the serial number of task and the total number of tasks, respectively. \( \text{cur}\_\text{particle}\_\text{fit} \) represents the fitness value of current particle. The smaller the value is, the better position of current particle is(according to Eq.15). Generally, the termination condition is the maximum iteration times or a good enough solution. If the termination condition is met, the solution will be obtained and the run is terminated.

4.2 Calculation Process of Start Execution Time of the Task’s Primary and Backup Copies

In the proposed of model this paper, the start time of primary copy should be as early as possible while the start time of backup copy should be as late as possible, so that it can provide enough laxity for corresponding task backup copy to adopt passive mode, primary copy and active backup copy to overlap as little as possible, and thus improve the utilization of network resource. Therefore, it is necessary to compute the earliest start time of a task’s primary copy and the latest start time of a task’s backup

\[
\text{Algorithm 1 DPSO-Based Task allocation Scheme} \quad \begin{array}{ll}
\text{Input:} & \text{Tasks,Particles} \\
\text{Output:} & gBest \\
& \begin{array}{l}
1: \text{for each particle } P_i \text{ do} \\
2: \quad pBest_i = \text{Generate}\_\text{initial}\_\text{position}(P_i); \\
3: \text{end for} \\
4: \text{for the } pBest \text{ of each particle } P_i \text{ do} \\
5: \quad gBest = \text{Max}(pBest_1,pBest_2...); \\
6: \text{end for} \\
7: \text{while the termination condition is not met do} \\
8: \quad \text{Let } j = 1; \\
9: \quad \text{while } j \leq m \text{ do} \\
10: \quad \quad \text{Select the task } t_j; \\
11: \quad \quad \text{calculate}\_\text{est}(t_j^P); \\
12: \quad \quad \text{calculate}\_\text{lst}(t_j^B); \\
13: \quad \quad \text{allocate}\_\text{ptask}(t_j^P); \\
14: \quad \quad \text{allocate}\_\text{btask}(t_j^B); \\
15: \quad \quad j + +; \\
16: \text{end while} \\
17: \text{for each particle } P_i \text{ do} \\
18: \quad \text{Calculate cur}\_\text{particle}\_\text{fit}; \\
19: \quad \quad \text{if } \text{cur}\_\text{particle}\_\text{fit} < \text{pBest}\_\text{fit} \text{ then} \\
20: \quad \quad \quad \text{update}(\text{pBest}_i); \\
21: \quad \quad \text{end if} \\
22: \quad \quad \text{if } \text{cur}\_\text{particle}\_\text{fit} < gBest\_\text{fit} \text{ then} \\
23: \quad \quad \quad \text{update}(gBest); \\
24: \quad \quad \text{end if} \\
25: \quad \text{end for} \\
26: \quad \text{if the termination condition is met then} \\
27: \quad \quad \text{output } gBest;\text{Break}; \\
28: \quad \text{else} \\
29: \quad \quad \text{for each particle } P_i \text{ do} \\
30: \quad \quad \quad \text{update}(P_i\_\text{position}); \\
31: \quad \quad \quad \text{update}(P_i\_\text{velocity}); \\
32: \quad \quad \text{end for} \\
33: \quad \text{end if} \\
34: \text{end while}
\end{array}
\]
4.2.1 Calculation Process of the Earliest Start Time of the Task’s Primary Copy

Assuming that there exists a task \( t_j \), the earliest start time of \( t_j^p \) which is allocated to node \( n_i \) can be calculated as follows: according to theorem 1, we scan each node’s idle time slot \([0, S_i^1], [f_1, S_i^2], [f_2, S_i^3], \ldots, [f_n, +\infty]\) from left to right, where \( S_i^1 \) and \( f_i \) represent the start time and finish time of the \( i \)-th task in the task queue of \( n_i \), respectively. For consistent expression, let \( f_0' = 0 \). If the first idle time slot \([f_0, S_{k+1}]\), which can meet \( \max(a_j, f_k) + e_t j, s_i \leq d_j \), is found, the earliest start time of \( t_j^p \) on node \( n_i \) would be noted as \( est_ji = f_k \), otherwise \( est_ji \) would be noted as \(+\infty\).

4.2.2 Calculation Process of the Latest Start Time of the Task’s Backup Copy

Assuming that there exists a task \( t_j \) is a primary copy or an active backup copy, update slot with \( est_ji \) and \( et_ji \) and then allocate \( t_j^p \) to the selected node and update its corresponding \( U^P(i, j) \) and \( UB(i, j) \) for next calculation convenience.

4.3 Allocation Process of the Task’s Primary Copy

In this section, we design a task allocation process for the task’s primary copy. Assuming that a task’s primary copy to be allocated is \( t_j^p \), the process considers task deadline constraint, as well as each node’s load, energy consumption and the failure ratio. Those factors are quantified and weighting accumulated by data standardization function. And then we design a utility function \( U^P(i, j) \) to measure comprehensive performance of each node executing \( t_j^p \) and allocate \( t_j^p \) to a superior node.

\[
U^P(i, j) = wt_1 \times UB(i, j) + wt_2 \times UE(i, j) + wt_3 \times UR(i, j)
\]

where \( wt_1, wt_2 \) and \( wt_3 \) are weight coefficient, \( UB(i, j) \) is the utility function of primary copy, the smaller the value of \( UB(i, j) \) is, the better \( n_i \) execute \( t_j^p \) comprehensively. \( UE(i, j) \) and \( UR(i, j) \) denote the load degree, energy consumption degree and failure ratio degree of \( n_i \), which executing \( t_j^p \) compared with other nodes which take part in \( t_j \). The current load bi, energy consumption \( ene_{ji} \) and failure ratio \( \lambda_i \) of \( n_i \) are mapped in the range of \( 0 \) and \( 0.5 \) by using a similar sigmoid function as data standardization function to realize \( UB(i, j), UE(i, j) \) and \( UR(i, j) \). The specific calculation formulas are shown as follows:

\[
UB(i, j) = \begin{cases} 
0, & \text{if } \big( b_{\max} - b_{\min} \big) = 0 \\
\frac{1}{e^{\frac{\lambda_{\max} - \lambda_{\min}}{\max_{\lambda - \min} + 1}}}, & \text{else} 
\end{cases} 
\]

(17)

\[
UE(i, j) = \begin{cases} 
0, & \text{if } \big( e_{\max} - e_{\min} \big) = 0 \\
\frac{1}{e^{\frac{e_{\max} - e_{\min}}{\max_{e - \min} + 1}}}, & \text{else} 
\end{cases} 
\]

(18)

\[
UR(i, j) = \begin{cases} 
0, & \text{if } \big( \lambda_{\max} - \lambda_{\min} \big) = 0 \\
\frac{1}{e^{\frac{\lambda_{\max} - \lambda_{\min}}{\max_{\lambda - \min} + 1}}}, & \text{else} 
\end{cases} 
\]

(19)

4.4 Allocation Process of the Task’s Backup Copy

Usually, the passive backup copy does not be executed, therefore only failure ratio and load should be considered to measure nodes utility for the passive backup copy, while failure ratio, load and energy consumption should be considered for the active backup copy. Therefore, we design another utility function, which can treat passive and active backup copy without difference. By using data standardization function, the performance can also be quantified and weighting accumulated to measure comprehensive performance of each node and allocate the backup copy to a better node.

According to theorem 2 and 3, the specific allocation process of the task’s backup copy is shown as follows:

\[
\text{Step 1: There exists a backup copy } t_j^B \text{. For each node } n_i \text{ which participates in } t_j, \text{ calculate the sum of } \text{lst}_j, \text{ the } \text{est}_j, \text{ and } \text{et}_j, \text{ to determine the perform node of backup copy and predict the finish time of } t_j^p \text{ on node } n_i . \text{ If the deadline constraint of } t_j^B \text{ can be met, } U^B(i, j) \text{ will be calculated according to formula (21), otherwise, } U^B(i, j) \text{ will be noted as } +\infty \text{ until all those nodes have been considered.}
\]
Step 2: Select a node with the minimum $U^B(i, j)$, allocate $t^B_j$ to the selected node and update the corresponding $U^B(i, j)$ and $UB(i, j)$;

$$UE(i, j) = \begin{cases} 0, & \text{if } (ene^\prime_{\text{max}} - ene^\prime_{\text{min}}) = 0 \\ \frac{ene^\prime_{\text{max}} - ene^\prime_{\text{min}}}{ene^\prime_{\text{max}} - ene^\prime_{\text{min}}} - 0.5, & \text{else} \end{cases}$$ (20)

$$U^B(i, j) = wt_1 \times UB(i, j) + wt_2 \times UE(i, j)$$
$$+ wt_3 \times UR(i, j)$$ (21)

where $wt_1, wt_2$ and $wt_3$ are weight coefficients, $U^B(i, j)$ is the utility function of backup copy. The smaller the value of $U^B(i, j)$ is, the better $n_l$ execute $t^B$ comprehensively. $ene^\prime_{\text{min}}$ and $ene^\prime_{\text{max}}$ denote the least and the most energy consumption of nodes which participate in $t_j$, respectively.

$ene^\prime_{\text{min}}$ and $ene^\prime_{\text{max}}$ can be calculated as follows:

$$ene^\prime_{\text{min}} = \min_{i=1}^n (x_{ji} \times S(t^B_j) \times ene_{ji} \times per_{ji})$$ (22)

$$ene^\prime_{\text{max}} = \max_{i=1}^n (x_{ji} \times S(t^B_j) \times ene_{ji} \times per_{ji})$$ (23)

### 5.1 A Simple Example of Our FTAOA Algorithm

Given a small-scale WSN with three sensor nodes $\{n_1, n_2, n_3\}$ whose failure probability are 0.01, 0.02 and 0.02. There are two tasks $\{t_1, t_2\}$ to be executed. Both two tasks’ arrival time is 0 and their deadlines are 3 and 4, where $ET = [1 2 1 1 2 3]$ and $ENE = [1 2 1 1 3]$. And the parameters of $wt_1, wt_2$ and $wt_3$ are set as 0.4, 0.4 and 0.2, respectively. Now we assume that the position matrix is $X = [1 0 1 1 0]$ for a particle.

Because both $x_{11}$ and $x_{13}$ are equal to 1, only $n_1$ and $n_3$ are allowed to execute $t_1$, and likewise, only $n_1$ and $n_2$ are allowed to execute $t_2$. For $t^B_1$, $UB(1) = UB(3) = 0$, $UE(1, 2) = UE(3, 1) = 0$, $UR(1, 2) = 0 < UR(3, 1) = 1/0 - 0.5$ and the value of $U^B(1, 1)$ is smaller, so $t^B_1$ is allocated to $n_1$ and $t^B_2$ occupies $n_1$’s time slot [0, 1]. Besides, $t^B_2$ is allocated to $n_3$, which occupies $n_3$’s time slot [2, 3] and adopts passive mode. For $t^B_2$, $UB(1, 2) = 0$, $UB(2) = 0 < UB(3, 1) = 0$, $UE(1, 2) = UE(2, 1) = 0$, $UR(1, 2) = 0 < UR(2, 1) = 1/0 - 0.5$, and $U^B(1, 2) = 0.4 \times UB(1, 2) + 0.2 \times UR(1, 2) > U^B(2, 1) = 0.4 \times UB(2, 1) + 0.2 \times UR(2, 1)$, so $t^B_2$ is allocated to $n_2$ and $t^B_2$ occupies $n_2$’s time slot [0, 2]. And then $t^B_1$ is allocated to $n_1$, because $n_0(p(t^B_1))$ should not be $n_0(p(t^B_2))$ according to property 1. Next, this algorithm considers another particle at this iteration until all particles would have been calculated, and then calculates the fitness value according to formula (15) for each particle and updates particles. It repeats above procedure until the termination condition is met and finally gets a superior allocation mode.

### 5. Simulation Results and Analysis

#### 5.1 Experimental Setup

Our simulation study is conducted for a WSN of $n$ nodes that are placed uniformly randomly in a rectangular region of 100 by 100 meters. We consider sensors which are out of energy or out of work suddenly as the failure node. Here, we consider the network is out of work once more than one fifth of sensors fail. The frequency of CPU of the computer we used in our experiment is 3.10 GHZ. Our simulation environment includes visual studio 2010 and matlab 7.0. In order to evaluate the performance of FTAOA, we compare FTAOA with QoS-aware fault-tolerant scheduling algorithm (QAFS), which is the latest proposed efficient fault-tolerance task allocation algorithm on heterogeneous clusters [6]. Meanwhile, to reveal performance improvements gained by backup copy overlapping technology, we also compare FTAOA with fault-tolerance task allocation algorithm without overlapping (FTAA). And the difference between FTAOA and FTA just lies in the fact that FTAOA does not consider the overlap of backup copies. Besides, to further study the optimization ability of PSO, we compare FTAOA with fault-tolerant task allocation with genetic algorithm (TAGA). Min-Min [10], [33] algorithm is a traditional heuristic approach on task allocation. To further verify the effectiveness of FTAOA, we extend Min-Min approach with P/B technology (name it EMM) and compare EMM with our proposed algorithm. Furthermore, in order to facilitate comparison and to be fair, we redesign QAFS without QoS requirement. And these algorithms are applied in fault-tolerant task allocation architecture of this paper. Generally, the parameters of simulation are set as follows: both values of $t_{ijk}$ and $ene_{ij}$ are selected from a uniform distribution in [200, 400]. $\alpha$, $\beta$, $\gamma$ and $\theta$ are set as 0.2, 0.2, 0.5 and 0.1. $wt_1, wt_2$ and $wt_3$ are set as 0.4, 0.4 and 0.2. $m_{iter}$, $w_{max}$ and $w_{min}$ are set as 200, 0.9 and 0.4. Both $c_1$ and $c_2$ are set as 2. $pxover$ and $pmutation$ are set as 0.8 and 0.15, respectively(crossover probability and mutation probability use in TAGA). In this paper, we compare FTAOA with FTA, QAFS, TAGA and EMM in the metrics shown in Table 3. But because EMM is extended from a traditional heuristic approach and it does not construct fitness function like formula (15), thus EMM is not involved in fitness comparison with other four algorithms.

<table>
<thead>
<tr>
<th>Metrics</th>
<th>The Meaning and Feature of Each Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deadline Missing Ratio [6], [12]</td>
<td>The ratio of the failure tasks and the total tasks. The smaller, the better.</td>
</tr>
<tr>
<td>Energy Consumption [12], [30]</td>
<td>The total energy cost of tasks. The smaller, the better.</td>
</tr>
<tr>
<td>Time Consumption [30]</td>
<td>The total time cost of tasks. The smaller, the better.</td>
</tr>
<tr>
<td>Fitness</td>
<td>The details can be seen in formula (15). The smaller, the better.</td>
</tr>
<tr>
<td>Reliability Cost [6]</td>
<td>The reliability of the network can be inferred from reliability cost. The smaller, the better.</td>
</tr>
<tr>
<td>Network Lifetime [15]</td>
<td>We use the algorithm running times until the network is out of work to represent the lifetime of network. The larger, the better.</td>
</tr>
</tbody>
</table>
### 5.2 Results and Analysis

#### 5.2.1 Parameter Selection

For the fitness function, the selection of weighted parameters is extremely important. Appropriate weighted factors can make the algorithm achieve better results. To validate the rationality of the parameter selection, we compare energy consumption and time consumption through setting several groups of weighted factors. The experimental results are shown in Table 4.

It can be seen from Table 4 that compared to other parameter settings, the values of factors we use in this paper can achieve better results in energy consumption and execution time consumption. The values of $\alpha$, $\beta$, $\gamma$ and $\theta$ are obtained by experiment more than once.

#### 5.2.2 Performance Impact of Task Deadline

First, we carry out a group of experiments in order to observe the performance impact of task deadline on FTAOA, FTAA, QAFT, TAGA and EMM. Here, the number of tasks is set as 200 and the deadline of tasks is distributed uniformly over $[100, 200]$, $[125, 225]$, $[150, 250]$, $[175, 275]$, $[200, 300]$, $[225, 325]$ and $[250, 350]$ for 7 times experiments.

Fig.4(a)-(e) show that the deadline missing ratio, network energy consumption, execution time and reliability cost of five algorithms are dropping with the increasing of task deadline time. FTAOA adopts passive backup copy overlapping technology to avoid unexecuted passive backup copies competing with follow-up tasks for network resource, and FTAOA can select the node with optimal comprehensive performance to execute tasks. Therefore, FTAOA outperforms QAFT and FTAA on deadline missing ratio, execution time consumption, network energy consumption and comprehensive performance. TAGA also adopts passive backup copy overlapping technology and selects the node with optimal comprehensive performance to execute tasks. Thus, it outperforms QAFT and FTAA on deadline missing ratio, execution time, network energy consumption and comprehensive performance as well. But TAGA is still worse than FTAOA on each index. Because TAGA weeds out the worst individual each time and is more likely to select individuals which have better fitness. Thus, many individuals will die out. On the contrary, FTAOA updates the population by using the best position experienced by each particle and the global best position experienced by the whole population. Hence, it keeps the population diversity better than TAGA and gets rid of local optimal situation that TAGA suffers from. QAFT does not consider comprehensive utility of nodes and just only considers to meet the deadline constraints, so that it allocates tasks to the node with the lowest failure ratio to improve network reliability and cannot develop the individual advantage of nodes. Thus, QAFT has the best performance on network reliability, and its deadline missing ratio is close to FTAOA and TAGA. Likewise, EMM does not sufficiently exploit comprehensive ability of nodes and only chooses the node which has the smallest execution time to perform a task. Thus, EMM has the best performance on total time consumption. Backup copy overlapping technology is not considered in FTAA, which makes the unexecuted passive backup copies hold an unnecessary space in the time slot of node and there may be abundant tasks waiting for network resource to be executed, so FTAA performs not well on every index. Because FTAOA adopts the passive backup copies overlapping technology to save network resources, it objectively provides more allocation mode opportunities for follow-up tasks. Thus FTAOA has the longest network lifetime as shown in Fig.4(f). Similar to FTAOA, TAGA has the second longest network lifetime. Moreover, EMM and QAFT do not optimize network energy consumption and balance network load, so they cause the load of some nodes extremely unbalanced and some nodes consuming much more energy, which leads to short lifetime.

#### 5.2.3 Performance Impact of Nodes

In this section, a group of experiments have been done to observe performance impact of different number of nodes. Without loss of generality, 400 tasks whose deadline randomly distributes over $[200, 400]$ are tested respectively in networks with 100, 125, 150, 175, 200, 225 and 250 nodes, and the remained energy of each node uniformly distributes over $[45000, 55000]$. The results are shown in Fig.5.

As shown in Fig.5(a)-(e), because more nodes and modes are available for task allocation, the deadline missing ratio, network energy consumption, execution time and reliability cost of five algorithms are dropping with the increasing of the number of nodes. Similar to the results in the last section, both FTAOA and TAGA outperform QAFT and FTAA on deadline missing ratio, execution time, network energy consumption and comprehensive performance. But there is still a gap between TAGA and FTAOA. QAFT has the best performance on network reliability and EMM has the best performance on execution time consumption. FTAOA has the longest network lifetime, and TAGA is the second.

#### 5.2.4 Performance Impact of Tasks

In order to observe performance impact of different number of tasks, without loss of generality, 150 nodes whose energy uniformly distributes over $[45000, 55000]$ are tested in networks with 200, 250, 300, 350, 400, 450 and 500 tasks, respectively. And the deadline of each task randomly distributes over $[200, 400]$.

As shown in Fig.6, due to limited network resource and node ability, more and more tasks cannot be finished before their deadlines with the increasing of task number, thus the deadline missing ratio, network energy consumption, and balance network load, so they cause the load of some nodes extremely unbalanced and some nodes consuming much more energy, which leads to short lifetime.

### Table 4: The results of different parameters setting

<table>
<thead>
<tr>
<th>Parameter($\alpha$, $\beta$, $\gamma$, $\theta$)</th>
<th>Energy Consumption</th>
<th>Time Consumption</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0.2, 0.2, 0.3, 0.2)</td>
<td>61881</td>
<td>43516</td>
</tr>
<tr>
<td>(0.25, 0.25, 0.25, 0.25)</td>
<td>62033</td>
<td>43804</td>
</tr>
<tr>
<td>(0.3, 0.3, 0.1)</td>
<td>62591</td>
<td>44137</td>
</tr>
<tr>
<td>(0.4, 0.2, 0.2)</td>
<td>63128</td>
<td>44342</td>
</tr>
<tr>
<td>(0.5, 0.1, 0.2, 0.2)</td>
<td>64542</td>
<td>43612</td>
</tr>
</tbody>
</table>
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2014.2386343, IEEE Transactions on Parallel and Distributed Systems

In this section, the rate of convergence of our proposed algorithm is investigated. Without loss of generality, 100 nodes whose energy uniformly distributes over [45000, 55000] are tested in networks with 250, 300 and 350 tasks, respectively. And the deadline of each task randomly distributes over [200, 400]. The value of initial iterative times and the step-size of these experiments are set as 5 and 15, respectively.

As shown in Fig. 7(a)-(c), we can find that after iterating fifty times approximately, FTAOA can achieve a proper solution. The algorithm begins to converge after iterating one hundred times approximately.

6 Conclusion

This paper presents a novel soft real-time task fault-tolerant allocation algorithm, FTAOA, in WSNs. In FTAOA, priority level is assigned to each task according to the idea of earliest deadline first, and the task with high priority will
be considered firstly. The proposed algorithm employs P/B technology and backup copy overlapping technology to achieve fault-tolerant mechanism. By taking task execution time, saving network consumption, balancing network load and improving deadline missing ratio and system reliability cost as optimization goals and employing a binary matrix encoding form, we construct a DPSO method to solve the task allocation problem and design a utilization function to evaluate comprehensive performance of nodes. To evaluate the performance of FTAOA, we conduct extensive simulations to compare FTAOA with FTAA, QAFT, TAGA and EMM. Simulation experiments show that the proposed algorithm is effective. In the future, we will focus on further improving fault-tolerant to reduce unnecessary redundancy by additionally employing active backup overlapping technology.

ACKNOWLEDGMENTS

This work is partly supported by the National Basic Research Program of China under Grant No.2009CB320503, the national Natural Science Foundation of China under Grant No. 61103175 and No.61401100, the Fujian Province Key Laboratory of Network Computing and Intelligent Information Processing Project under Grant No.2009J1007, the Key Project of Chinese Ministry of Education under Grant No.212086, the Fujian Natural Science Funds for Distinguished Young Scholar under Grant 2014J06017, the Program for New Century Excellent Talents in Fujian Province University under Grant JA13021, the Fujian Province High School Science Fund for Distinguished Young Scholars under Grant JA12016 and, the Natural Science Foundation of Fujian Province under Grant No. 2013J01235.

REFERENCES


Wenzhong Guo received the B.S. and M.S degrees in Computer Science from Fuzhou University, Fuzhou, China, in 2000 and 2003, respectively, and the Ph.D degree in Communication and Information System from Fuzhou University in 2010. He is currently a full professor with the College of Mathematics and Computer Science at Fuzhou University. His research interests include intelligent information processing, sensor networks, network computing, and network performance evaluation. Currently, he is the vice-director of the Network Computing & Intelligent Information Processing Lab, which is a key Lab of Fujian Province, China. He is a member of ACM, a senior member of China Computer Federation (CCF).

Jie Li (M’94-SM’04) received the B.E. degree in computer science from Zhejiang University, Hangzhou, China, the M.E. degree in electronic engineering and communication systems from China Academy of Posts and Telecommunications, Beijing, China. He received the Dr. Eng, degree from the University of Electro-Communications, Tokyo, Japan. He has been with University of Tsukuba, Japan, where he is a full Professor. His research interests are in mobile distributed multimedia computing and networking, OS, network security, modeling and performance evaluation of information systems. He received the best paper award from IEEE ICME, 2010, and the best paper award from IEEE MASS.
Guolong Chen received the B.S. and M.S degrees in Computational Mathematics from Fuzhou University, Fuzhou, China, in 1987 and 1992, respectively, and the Ph.D degree in Computer Science from Xi’an Jiaotong University, Xi’an, China, in 2002. He is a full professor with the College of Mathematics and Computer Science at Fuzhou University. His research interests include intelligent information processing, wireless networks, network security, and cloud computing. Currently, he leads the Network Computing & Intelligent Information Processing Lab, which is a key Lab of Fujian Province, China. He is a member of ACM, a senior member of China Computer Federation (CCF).

Yuzhen Niu received the B.S. and Ph.D. degrees in computer science from Shandong University, Jinan, China, in 2005 and 2010, respectively. From 2010 to 2012, she was a Postdoctoral Researcher in the Department of Computer Science at Portland State University, Portland, OR. She is a full professor with the College of Mathematics and Computer Science at Fuzhou University. Her research interests are in the areas of computer graphics, multimedia systems, video networking, and sensor networks.

Chengyu Chen received his B. S. and M.S degrees from Fuzhou University, Fuzhou University in 2010 and 2013, respectively. His research interests are in sensor networks and green networking.