On the Handling of Node Failures: Energy-Efficient Job Allocation Algorithm for Real-time Sensor Networks

Hamid Karimi*, Mehdi Kargahi*,** and Nasser Yazdani*

Abstract—Wireless sensor networks are usually characterized by dense deployment of energy constrained nodes. Due to the usage of a large number of sensor nodes in uncontrolled hostile or harsh environments, node failure is a common event in these systems. Another common reason for node failure is the exhaustion of their energy resources and node inactivation. Such failures can have adverse effects on the quality of the real-time services in Wireless Sensor Networks (WSNs). To avoid such degradations, it is necessary that the failures be recovered in a proper manner to sustain network operation. In this paper we present a dynamic Energy efficient Real-Time Job Allocation (ERTJA) algorithm for handling node failures in a cluster of sensor nodes with the consideration of communication energy and time overheads besides the nodes' characteristics. ERTJA relies on the computation power of cluster members for handling a node failure. It also tries to minimize the energy consumption of the cluster by minimum activation of the sleeping nodes. The resulting system can then guarantee the Quality of Service (QoS) of the cluster application. Further, when the number of sleeping nodes is limited, the proposed algorithm uses the idle times of the active nodes to engage a graceful QoS degradation in the cluster. Simulation results show significant performance improvements of ERTJA in terms of the energy conservation and the probability of meeting deadlines compared with the other studied algorithms.

Keywords—Failure Recovery, Job Allocation, Quality of Service, Real-Time Scheduling, Wireless Sensor Network

1. INTRODUCTION

Real-time networked and distributed embedded systems such as wireless sensor networks (WSNs) have increasingly become popular over the last decade [5, 14]. Examples of such systems are platforms for executing collaborative applications such as target tracking and infrastructure monitoring. A collaborative application consists of a number of tasks which are allocated to available sensor nodes of a cluster to fulfill a predetermined real-time mission [24, 16, 1]. Another example of challenging applications in this area is video sensor networks. Since multimedia processing generally involves computation-intensive operations, a single sensor

※ This research was in part supported by a grant from IPM. (No. CS1388-4-07).
Manuscript received June 16, 2010; accepted August 23, 2010.
Corresponding Author: Hamid Karimi
* School of Electrical and Computer Engineering, College of Engineering, University of Tehran, Iran ({hkkarimi, kargahi, yazdani}@ut.ac.ir)
** School of Computer Science, Institute for Research in Fundamental Sciences (IPM)

node may not be able to complete the respective jobs [8].

Almost all of the nodes in a sensor network suffer from the limitations of energy resources. The energy source in each sensor node is limited to the initial battery charge. Replenishing the battery charge is infeasible and so costly that it may overcome any benefits drawn from the WSN. Consequently, most of the research around WSNs focuses predominately on energy conservation problems. One typical mission-critical real-time sensor network has the following two characteristics: The first characteristic is guaranteeing the desired timing constraints and the second one is continuing the specified mission in spite of hardware or software failure. For example, in Embedded Control Systems (ECSs), for diverse applications such as avionics, automated manufacturing, and air-traffic control, the need for dependable systems that deliver services in a timely manner has become crucial. Thus, a fundamental requirement of ECSs is to complete all real-time tasks within their specified timing constraints even in the presence of faults [25]. Accordingly, fault-tolerance (and/or fault-recovery) techniques as well as real-time resource management (or scheduling) algorithms should be used together to be able to guarantee the desired QoS in such systems.

In this paper, we consider a single-hop cluster of a WSN consisting of homogeneous sensor nodes. A cluster executes an application which is either assigned during the network setup time or remotely distributed by base stations during the network operation. Each application consists of a number of tasks that have been allocated and scheduled in an offline manner among the sensor nodes. Once assigned, the tasks are independently executed on the working nodes within the cluster. Upon the failure of one of the nodes in the cluster (e.g., due to the corruption of the node or exhaustion of the battery), some recovery procedure should be followed to avoid of QoS degradation (The QoS measure that we have considered in this paper is the number of jobs required to meet their deadlines). This is done through a task reallocation algorithm, which dynamically assigns the jobs of the failed tasks (previously located on a failed node) to the remaining failure-free nodes in the cluster. This algorithm tries to avoid excessive activation of the sleeping nodes to conserve as much energy as possible in the cluster. In summary, the algorithm proposed in this paper includes the following main properties:

- It guarantees that the QoS of the overall application will be held at the same acceptable level even after an overloaded node failure,
- It tries to heuristically activate the minimum number of sleeping nodes in the cluster to conserve more energy,
- The algorithm tries to accrue the maximum possible QoS in the cases which, after failure, there are not enough sleeping nodes in the cluster to be activated to satisfy the target QoS,
- The algorithm uses a job admission control called Link Energy-aware Job Admission Control (LEJAC). This algorithm selects a node that can guarantee meeting the deadline of a job and yields the least energy consumption induced by corresponding messages. In other words, the proposed algorithm is also aware of the energy consumption of the communication links among the nodes which are asymmetric in nature.

This paper is willing to develop a job allocation algorithm for real-time sensor networks where reliability and performance are of high importance. In such systems, some kind of quality of service guarantee is desired and usually it is not acceptable to lose jobs when a node failure occurs.

The rest of this paper is organized as follows: In the next section we discuss the related works. Section 3 describes the system, task and energy consumption models. In Section 4, we describe a resource augmentation method which is used as a basic idea in our algorithm. Section 5 proposes our job allocation algorithm, and presents an online job admission control in detail. Section 6 discusses experimental results. Finally, we conclude the paper in Section 7.

2. RELATED WORK

Task allocation has been well studied in the area of distributed systems [24, 26]; however, in many studies in the literature of scheduling in real-time WSNs, the effects of fault occurrences in the systems are ignored, despite the fact that they have big impact on the performance and QoS of the systems. EAS [7] statically schedule communication and computation onto heterogeneous Network-on-Chip architectures under real time constraints. They focus on the architectures interconnected by 2D mesh networks with XY routing schemes. CoRAl [6] is an online task scheduling mechanism proposed to allocate network resources to tasks of periodic applications in WSNs. CoRAl does not address the mapping of tasks to sensor nodes, and energy consumption is not explicitly discussed.

A TCP oriented distributed task mapping approach is introduced for mobile ad hoc networks in [4]. DFuse [11] is a data fusion task mapping mechanism for WSNs. None of these solutions tolerate node failures. Task mapping and scheduling heuristics are presented in [19] for heterogeneous mobile ad hoc grid environments. EcoMapS [23] presents a generic task mapping and scheduling solution for single-hop clustered wireless sensor networks. Based on realistic energy models for computation and communication, it aims to provide energy consumption guarantees with minimum schedule lengths. In [18], the RTFTRC algorithm has been proposed. The objectives of this algorithm are to minimize the schedule length and maximize the reliability of the system. At the beginning, RTFTRC allocates the main and backup copies of tasks to the sensor nodes, leading to the maximum possible increase in the system reliability. This method is expensive and greatly increases overhead costs of the system.

None of the above studies consider the simultaneous constraints resulting from energy limitations, fault occurrences, and QoS guarantees in WSNs with overloaded nodes.

3. SYSTEM, APPLICATION AND ENERGY CONSUMPTION MODEL

3.1 System Model

The system consists of a set of n homogeneous sensor nodes shown with S_i, i=1,..,n. The sensors are grouped into 1-hop clusters with a specific clustering algorithm. The networked cluster can be represented as a graph of nodes and their point-to-point links. In the system, a sensor node is modeled as a vertex. There exists a weighted edge between two vertices if they can communicate with each other. Each node in the system has an energy consumption rate measured by Joules per unit of time. With respect to energy conservation, each network link is characterized by its energy consumption rate, which heavily relies on the link's transmission rate.

In this paper, $E = \{e_1, e_2 \ldots, e_m\}$ is a set of edges representing communications among the cluster-heads and the other nodes. The Communication time spent in delivering a message (contain-

ing job specification) from the cluster-head to the target node (for execution of a failed job) is T_{comm} = *datasize(m) / b* (S_u,S_v), where *datasize (m)* is the data amount of message *m* to be transmitted and *b*(S_u,S_v) is the transmission rate of the link that connects the nodes S_u and S_v. For the sake of simplicity, we assume that all nodes of the cluster are fully connected with a dedicated communication system. Without loss of generality, we apply the remaining of our discussions within a single cluster.

3.2 Application Model

A set of *m* independent periodic real-time tasks, $\{T_1, T_2, \ldots T_m\}$, are considered. A periodic task is a sequence of jobs released at constant intervals (called the period). Each job in a task is considered a processing request. It is associated with an absolute deadline by which the job should be completed. The period of T_i is denoted by P_i, which is assumed to be equal to the *relative deadline* (the difference between a job's absolute deadline and its release time, namely the start of the current period of T_i) of each job. This means that each instance of a task must complete its execution before the release of the next instance of the task. Each instance of the task has a computation time, illustrated by C_i.

3.3 Energy Consumption Model

A simplified sensor node energy consumption model is used here as a metric for evaluating energy consumption. A wireless node can take three different states with regard to energy: The **Sleep** state consumes the lowest power. The processor core is in sleep mode and all hardware components are deactivated. The power value (joules per second) for this state is ψ_S.

In **idle** state no specific task is performed, besides background OS processes [15]. The processor core may or may not be in its "idle" mode; this is controlled by the operating system. This node uses power value ψ_I.

When **active**, the node performs specific, finite–duration tasks involving the processor core possibly in cooperation with other hardware components [15]. In this state, the power value is ψ_K. In formulating energy consumption; we assume that ***t*** is the time duration that a sensor node is involved in the system. $K(t)$ is the total time which sensor nodes are active. S(t) and I(t) also have similar definitions for sleep and idle modes, then we have :

$$K(t) = \sum_{i=1}^{n} \sum_{j=1}^{t} K_{ij} \tag{1}$$

$$I(t) = \sum_{i=1}^{n} \sum_{j=1}^{t} I_{ij} \tag{2}$$

$$S(t) = \sum_{i=1}^{n} \sum_{j=1}^{t} S_{ij} \ , \tag{3}$$

where *n* is the number of nodes which exist in a cluster. Thus, the total energy consumption of the cluster is denoted as E_{comp} which is:

$$E_{comp} = E_K + E_S + E_I \,, \tag{4}$$

where $E_K = K(t) \times \psi_K$, $E_S = S(t) \times \psi_S$ and $E_I = I(t) \times \psi_I$.

Communication energy can also be calculated as:

$$E_{comm} = ECL_{u,v} \times T_{comm} \,, \tag{5}$$

where $ECL_{u,v}$ is the energy consumption rate of the link between nodes of S_u, S_v and T_{comm} is the data transmission time. Note that the energy consumption rate of a network link depends on the transmission rate of the link. The energy consumption rate of the network links can be modeled by an $n \times n$ matrix $ECL = (ECL_{u,v})$, where $ECL_{u,v}$ is the energy consumption rate function of the link between the nodes of S_u and S_v.

Upon one node failure in the cluster (e.g., due to the corruption of the node or exhaustion of the battery), the initial schedule of the node becomes invalid. In such situations, jobs of the failed node may miss their deadlines. Thus, a recovery procedure is needed to avoid QoS degradation in the cluster. Some recovery procedure should dynamically allocate the jobs of the failed tasks (which previously were located on the failed node) to the remaining failure-free nodes in the cluster.

Since the initial (offline) schedule is lost, jobs of the failed node will be unknown to the recovery procedure till their release times. Thus, an online scheduling algorithm should be used to handle failures, and to execute jobs in the other available nodes. Online scheduling is not an optimal algorithm under overload conditions while offline scheduling is optimal [3]. To overcome this problem and guarantee the system QoS after node failure, we exploit a resource augmentation technique to generate an optimal on-line scheduling [10, 12, 9, 17]. This technique is described in the next section.

4. BASIC RESOURCE AUGMENTATION TECHNIQUE

For job scheduling in real-time systems, two types of algorithms are used:

- Offline scheduling algorithms
- Online scheduling algorithms

In on-line scheduling, the jobs arrive over time, and the online scheduler must make scheduling decisions without knowledge of the future. A typical example of online algorithms is the Earliest Deadline First (EDF) algorithm, which has been widely used in many real-time systems (see [20] for a survey). From a theoretical viewpoint, EDF is optimal for scheduling under-loaded systems, i.e., whenever there exists a schedule meeting the deadlines of all jobs released, EDF can always do so [20]. However, in the case of overloading in the system, no online algorithm can compete against its offline adversaries regarding the performance measures [3].

In recent years, a plausible approach to improving the performance guarantee for online scheduling (without restricting the inputs) is to allow the online scheduler to use more processors than the offline adversary. The use of more processors in online scheduling is meant to compensate for the online scheduler's lack of future information. Literature [10] proposed an online scheduling algorithm called N-EDF-Plus which is a 3-processor scheme optimal for online scheduling jobs. The paper proves that to schedule any task set T with uniform *value density*, the safe processing time produced by N-EDF-Plus is no less than the total processing time of the tasks completed on an optimal off-line algorithm using one processor. The value density of a job is defined to be its value divided by its processing time, and the *importance ratio k* of a system is the ratio of the largest possible value density to the smallest possible one. When $k = 1$ (uniform value density), it means that every job has a value proportional to its computation time.

5. PROPOSED ALLOCATION ALGORITHM

Guaranteeing quality of services (QoS) in sensor networks depends critically on tolerance of node failures. Nodes may fail due to several reasons, including energy exhaustion, material fatigue, environmental hazards or deliberate attacks. In order to guarantee QoS in instances of node failure, we present an Energy-Efficient Real-Time Job Allocation (ERTJA) algorithm which reallocates the jobs of the failed node to the other nodes in the same cluster. We try to handle node failure, and prevent QoS degradation while improving the energy conservation of the cluster in case of failure. To optimize performance, in the trade-off between QoS-guarantee and energy conservation, we employ a novel job admission control algorithm which is executed in the cluster-head and chooses a sensor node for the execution of the failed node jobs. Details of this algorithm are described in the following section.

5.1 Algorithm Description

ERTJA exploits the N-EDF-Plus algorithm proposed in [10] to guarantee QoS and obtaining the optimum value (optimum number of jobs that meet their deadlines) for the jobs of the failed node. ERTJA is a modified version of N-EDF-Plus which follows the steps of the latter algorithm in a virtual manner. N-EDF-Plus is a 3-processor optimal algorithm which is used for online scheduling of jobs with uniform values. It uses three empty (sleeping) nodes for obtaining some values identical to an off-line optimal schedule. The three additional nodes in N-EDF-Plus are used to compensate for the lack of future information in the offline schedule [10]. Requesting three sleeping nodes from the system is costly, and is impossible most of the time due to the resource constraints of such systems. Thus, this algorithm may not be used directly in sensor networks. On the other hand, ERTJA virtually provides three sleeping nodes and tries to use the active nodes rather than allocating jobs to the new nodes. Using virtual nodes, the ERTJA algorithm saves more energy than N-EDF-Plus in instances of node failure. When a node failure occurs, ERTJA is executed on the cluster-head and dynamically assigns the jobs of the failed node to the other available nodes in the cluster. ERTJA tries to minimize the energy consumption of the cluster through the minimum activation of sleeping nodes. On the other hand, to conserve energy, nodes should remain sleeping as much as possible.

In case of a node failure, ERTJA begins with three initial virtual nodes, namely S_e, S_d and S_u.

While one job J of a failed task is released, it will be admitted by S_e if possible. Virtual node S_e schedules jobs using EDF-AC. The EDF-AC denotes EDF enhanced with a simple form of admission control. Upon release of a job, it must pass a test to get admitted for EDF scheduling. The test simply checks whether the new job together with the previously admitted jobs can all be completed by their deadlines using EDF. In this regard, once S_e admits a job, the job is guaranteed to be completed by its deadline. After admission of J by S_e, ERTJA uses an online job admission control algorithm called LEJAC for mapping the job to an available node (working or sleeping) in the cluster (the LEJAC algorithm which is used to replace a virtual node with a real working or sleeping node is described in detail in the Subsection 5.2). If job J is rejected by S_e, it is put into a common pool shared by other virtual nodes. Whenever S_d is idle, it removes the job with the latest deadline from the pool and calls the LEJAC algorithm. If a job J in the pool has never been picked by S_d, its slack time will eventually become zero and it is then said to be *urgent*. In this case, the algorithm tries to schedule J by the virtual node S_u. Also, in this case, the LEJAC maps the job onto a real node. The ERTJA algorithm is described in detail in Fig. 1.

```
Se, Sd, Su are 3 virtual node.
Initialization:  P ← ∅
 When job J is released:
        if Se can admit J
             Se admits J to its EDF queue
             Call LEJAC (J)
        else if  Sd is idling
              Sd runs J
              Call LEJAC (J)
        else
                 P ← P ∪  {J}
 When Se completes a job J:
        Se removes J from its EDF queue
        Se schedules the job with the earliest deadline
        Call LEJAC (J)
 When Sd completes a job:
        if  P ≠ ∅
             Let J  be the job in P with the latest deadline
             P ← P − {J}
             // if  Su is running J, it becomes idle
            Sd runs J
            Call LEJAC (J)
 When J ∈  P becomes urgent (slack time becomes zero):
        if Su is idling
             Su runs J
             Call LEJAC (J)
        else
              Let J´  be the urgent job other than J in P with
              the earliest deadline
              Let Su be the processor running J´
                 // Su exists; otherwise J´  would miss its deadline
        if d(J) > d(J´ )
               P ← P − {J´ };
               Su runs J;
               Call LEJAC (J)
        else
              P ← P − {J}
```

Fig. 1. The ERTJA Algorithm

5.2 Link Energy-aware Job Admission Control (LEJAC)

In this section, we present a novel admission control scheme, which can achieve near optimal performance for a mixed set of periodic tasks and a number of new jobs with hard deadlines. We attempt to provide deadline guarantees via admission control for both of the periodic tasks and new jobs under EDF scheduling. The proposed admission control scheme, namely LEJAC, is based upon the slack stealing method, introduced by Thuel and Lehoczky [13, 21, 22].

Using slack stealing, the LEJAC provides a method for guaranteeing and scheduling new jobs with periodic tasks in the sensor nodes. With complete knowledge of the nodes' execution and future periodic requirements, the LEJAC can determine whether there is adequate time in the node to admit a new job. For the reallocation of jobs of the failed node(s), the LEJAC is executed in the cluster-head, and gathers information of the periodic tasks in the working nodes (under-loaded nodes) of the cluster. In order to maximize energy conservation, the LEJAC selects the node with the minimum ECL. The LEJAC, based on the knowledge of the node with respect to its task's information, creates two tables called T-EDF and T-DWP for each of the working nodes.

In practice, T-EDF and T-DWP tables may be a character string for a number of periodic tasks or may be an array of integers in general. The T-EDF table of each node is created based on the sequence of the executing task set of the node when sorted according to EDF. The T-DWP table is also created based on the assignment rule proposed by Deadline-Wise Pre-assignment (DWP) scheduling. In the DWP method, each task's execution is postponed toward its deadline as long as it does not miss its deadline. Using the above mentioned tables as well as slack information, the LEJAC determines whether there is enough slack to support the extra required resources in a working node. If there is adequate slack, then the job is mapped to the node. Otherwise, the job is checked for admission to another node. After assigning a job to one of the working nodes and sending a *J* specification message to the chosen node, the T-EDF and T-DWP tables of the node are updated. Therefore, the LEJAC is used to ensure that (1) only those jobs that will be complete by their deadline are admitted, (2) a job that is assigned to a node can only consume the slack in that working node and does not interfere with any previously assigned task(s) on the same node, and (3) the selected node has the minimum link energy consumption.

Example. Suppose that a node has a periodic task set with two tasks, T1 and T2, having the computation requirements, C1=1and C2=2, and the periods, P1=4 and P2=6, respectively. These tasks are to be pre-assigned over a single hyper-period, H=12.The T-EDF and T-DWP tables for the periodic task set given in this example are as the following character strings:

T-EDF= "122010221000"
T-DWP="000122010221"

The length of each table is equal to the length of the node task's hyper-period in time units, where '0', '1', and '2' respectively denote an empty time unit (slack), a time unit used by T1, and a time unit consumed by T2. With this simple data structure, all the execution start points and computation requirements of the periodic tasks can be fully represented in addition to the slacks which can be used for new admitting jobs.

Two kinds of special computation counter are extracted for periodic tasks from the tables:

1- Cumulating all computation processing completed (*CP*) which is extracted from the T-EDF table.
2- Cumulating the entire computation requirement (CR) which is retrieved from the T-DWP table.

To provide a clearer slack identifying mechanism for new jobs, we need to formalize these to a slack discriminant that can be used for distinguishing slacks at the T-EDF and T-DWP tables. The resulting slack discriminant for the LEJAC algorithm is:

$$S(D_J) = D_J - (R_J + T_{comm}) - (CR - CP), \tag{6}$$

where R_J, D_J denotes the release time and deadline of the new job and T_{comm} is the requirement time for sending a message to a node. *CP* and *CR* respectively denote all the computation processing done until the release time of the new job (the current scheduling time) and the entire computation requirement until the deadline of the new job. In this regard, the job is admitted when $S(D_J) \geq C_J$ (C_J is the computation requirement of the new job). The details of the LEJAC are depicted in Fig. 2.

Suppose that job *J* is released at instant 6 (R_J=6), its deadline is 9 (D_J=9) and its computation time is 2 (C_J=2). To allocate job *J* to one of the working nodes, the LEJAC algorithm calculates computation requirement (*CR*) from instant 1 to D_J in T-DWP table (*CR=4*). Then, it calculates computation processing done from instant 1 to $R_J + T_{comm}$ (we assumed that T_{comm}= 1) in T-EDF table (*CP=4*). Consider the above calculated T-EDF and T-DWP tables. We have:

$$S(9)= 9 – (6 + 1) – (4 – 4).$$

Therefore, since $S(9) \geq C_J$, the job is admitted.

```
Initialization: flag_admission ← false
Sort Si's respect to ECL
Foreach Si in set of working nodes
    If (DJ ≥ RJ + CJ + Tcomm)
        Build T-EDF table
        Build T-DWP table
        Calculate CR from T-DWP table
        Calculate CP from T-EDF table
        Calculate S(DJ) = DJ – (RJ + Tcomm) - (CR – CP)
         // S(DJ) is number of slack until time DJ
        If S(DJ) ≥ CJ // CJ is Computation requirement J
            Si admit J;
            Cluster-head send J specification message to Si
            Update T-EDF & T-DWP of Si
            flag_admission=true; break;
If not (flag_admission)
        Find Sk with minimum ECL
      Sk in set of sleeping nodes admit J
```

Fig. 2. The LEJAC Algorithm

6. SIMULATION RESULTS

A simulator based on the system and application models presented in Section III was developed to evaluate the performance of the ERTJA algorithm. We compare the results of this algorithm with respect to those of N-EDF-Plus and EcoMapS. Through our simulations we investigate the efficiency of the algorithms with respect to average energy consumption, probability of meeting deadlines and average number of sleeping nodes which are activated to handle possible failures. All simulations are run on a P-IV 2.26 GHz processor with 2GB of RAM. Simulation software was developed using the Microsoft Visual Studio 2008 using the C# programming language. We assume that there are 10 homogeneous sensor nodes in a single-hop cluster. The initial energy of each node is considered as 6000 mJ. The power consumption of each node is shown in Table 1. Hence, all results provided in this section are averaged over one hundred experiments per endpoint. The relationship between energy rate and transmission rate is 100 mW at 100 Kbps, which means that the time and energy cost for transmitting 1 bit are around 10 μs and 1 μJoule [2].

The task set considered in our simulations consists of 20 independent tasks with relative deadlines equal to their respective periods. Without loss of generality, we assume that the release time of the first job of all tasks is identical and equal to 0 (All the tasks are in phase and their phase is equal to zero). Random number generators were used to generate the task period time and the task execution (service) time. The tasks period time are uniformly distributed between [10, 50] while tasks execution time follow a uniform distribution between [1, 5]. To investigate the effects of the proposed job allocation policy and measure its performance, two tests, namely Case 1 and Case 2 are considered. For every test (Case 1&2), we have carried out one hundred experiments. In each of experiments, random number generators are used to produce different task sets. Also for each of the tests, we considered three different policies for initial task allocation to the available nodes in the cluster (task pre-allocation). In these policies, tasks are allocated to the nodes until utilization of the nodes exceeds no further than the acceptable utilization of the policy. These utilizations are listed in Table 2.

Typically, these policies are used in the various WSNs for the sake of load balancing and lifetime extension. The tested scenario considers failure occurring in an overloaded node that exists in the cluster. (It is possible that some nodes would be overloaded due to the deficiency of the allocation algorithm or system defects). In Case 1, we assume that after running each pre-allocation policy A, B and C, an overloaded node with a load of about 150% would exist in the cluster. The tasks which are assigned to the overloaded node are scheduled in an offline manner to maximize the attainable value of the tasks. However, the remaining nodes follow the EDF algorithm, which is a well-known optimal scheduling algorithm for under-loaded nodes. In each of the tests (Cases 1&2), N-EDF-Plus and ERTJA algorithms are executed to handle the node failure and allocate the respective failed tasks to the other available nodes in the cluster. Suppose that the cluster has reached its steady state behavior, when it has passed a 1000-second time

Table 1. power consumption of wireless sensor network

Mode of Sensor	Ratio
Sleep	0.001
Idle	0.8
Active	1

Table 2. acceptable utilization of policies

Policy	Utilization
Policy A	30%
Policy B	50%
Policy C	80%

period. We suppose that the failure has occurred at the most damaging time during a hyper period, i.e., at the start of this period. Fig. 3 depicts the efficiency of the two indicated algorithms on the energy consumption of the cluster after this failure (In all the experiments, we have normalized the consumed energy with respect to the time unit. In other words, we show the average power usage of the system in mWatts, which was obtained through the calculation of the respective mean value of 100 experiments).

The N-EDF-Plus is an optimal online scheduling algorithm formulated especially for overloaded nodes. However, it consumes unnecessary energy because it does not make use of the available capacity of any working nodes of the cluster. On the other hand, the ERTJA algorithm uses the working (under loaded) nodes, and tries to allocate the jobs of the failed node to them. As can be observed in Fig. 3, the energy conservation of ERTJA outperforms that of N-EDF-Plus up to 27.2% on average.

In Fig. 3, the related curves of both the algorithms in policy C are seen to be under the curves of the A and B policies. The reason is that more tasks are allocated to each node in this policy, consequently activating a lower number of nodes as well as consuming less energy. As it can be observed, the curve of policy C in the ERTJA algorithm is overlapped with the curve of policy B. This is due to the reason that when a node failure occurs in policy C, other nodes have not adequate idle time for the failed node tasks. Consequently, more sleeping nodes should be activated in this policy to avoid QoS degradation.

In Case 2, to observe the system behavior in the case of failure of an extremely overloaded

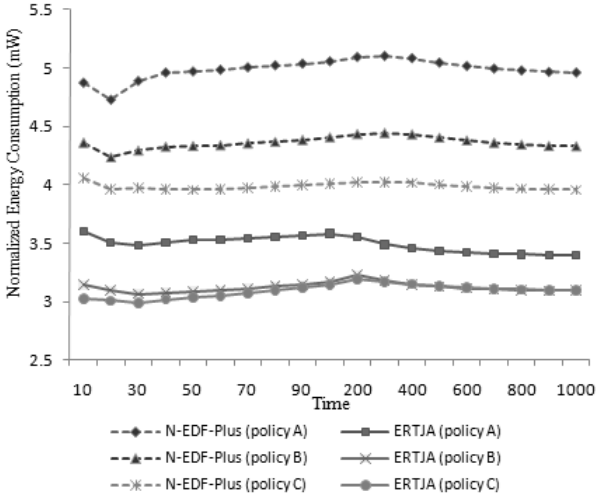

Fig. 3. Normalized energy consumption of N-EDF-Plus and ERTJA (Case 1)

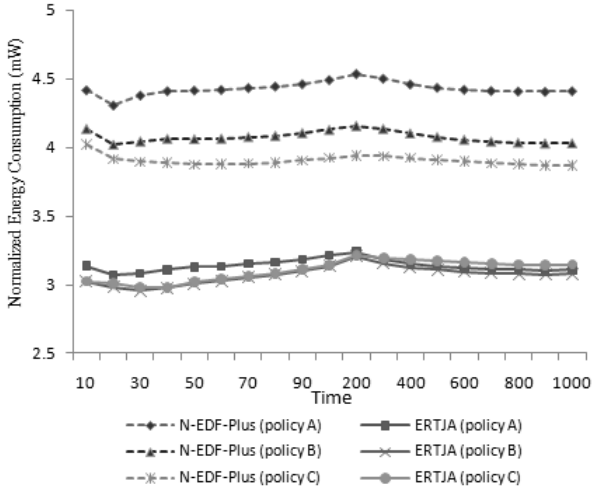

Fig. 4. Normalized energy consumption of N-EDF-Plus and ERTJA (Case 2)

node, the node workload is considered to be about 180%. As shown in Fig. 4, the ERTJA can save up to 23.9% of energy consumption on average compared to N-EDF-Plus.

Fig. 5 and Fig. 6 show the average number of nodes that are added for handling the node failure while guaranteeing the cluster's QoS by the two algorithms in Case 1 and Case 2. It should be noted that for better analysis, we excluded the failed node in calculating the average number of added nodes (before and after node failure). As it can be seen in Figs. 5 and 6, the curves of the three policies A, B, and C related to N-EDF-Plus are overlapped. This is due to the fact that this algorithm, undoubtedly, turns on the same number of the sleeping nodes in different policies.

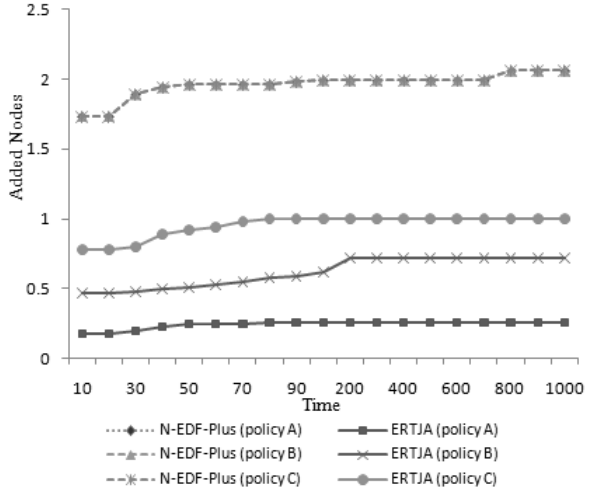

Fig. 5. Number of added nodes in N-EDF-Plus and ERTJA (Case 1)

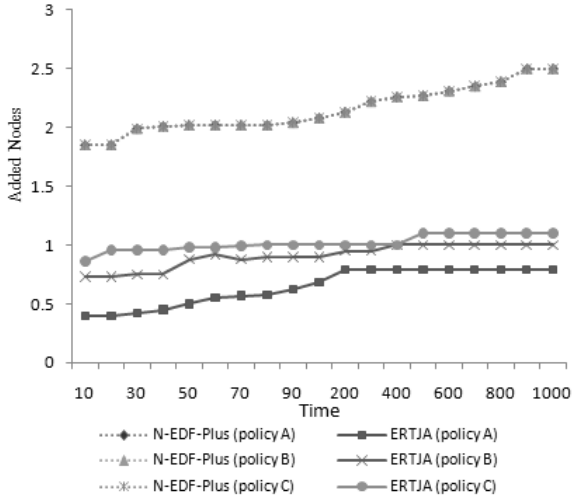

Fig. 6. Number of added nodes in N-EDF-Plus and ERTJA (Case 2)

According to Figs. 5 and 6, the N-EDF-Plus algorithm needs 2 or 3 sleeping nodes for guaranteeing the cluster's QoS and failure handling while the ERTJA algorithm needs only 1 sleeping node on average. Since in many situations, there may not exist enough sleeping nodes to be awakened, ERTJA encounters considerable preference over N-EDF-Plus. According to our simulation results, ERTJA is more efficient than N-EDF-Plus, and it can better be applied to resource constrained systems with possible failures.

In the following, we compare our proposed algorithm to EcoMapS [23] which may be the closest algorithm among the related works to ERTJA. EcoMapS is a task allocation and scheduling algorithm for energy-constrained applications in WSNs which works in the scope of a cluster, similar to ERTJA. The main goals of EcoMapS are to minimize the schedule length subject to some energy consumption constraints. In instances of node failures, EcoMapS overcomes the failure through a recovery phase and allocates tasks of the failed node to one of the other nodes existing in the cluster. It first tries to find an idle node (sleeping node) for the allocation. If this trial does not succeed, the algorithm tries to allocate the tasks to the node with the minimum load.

The experiments are presented for a comparative study between the performance of ERTJA and EcoMapS algorithms in the presence of one node failure. The results are presented for both Cases 1 and 2, introduced above. In other words, through our simulations, we investigate the efficiency of the algorithms with respect to the average energy consumption and probability of meeting deadlines in the presence of the failure. It should be noted that our previous assumptions for the case of comparison to N-EDF-Plus are also valid for the further experiments. To have a more extensive set of simulations, we consider two different conditions for the status of the cluster: Condition 1, in which the cluster has at least one sleeping (idle) node, and Condition 2, where the cluster has no sleeping node.

Fig. 7 depicts the efficiency of the two aforementioned algorithms on the energy consumption of the cluster in Condition 1 of the Case1 scenario after one node failure. As can be observed in

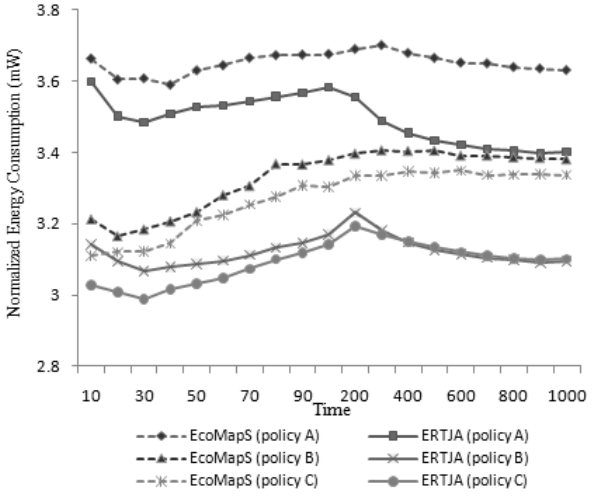

Fig. 7. Normalized energy consumption of EcoMapS and ERTJA (Condition 1 of Case 1)

the figure, the energy conservation of ERTJA outperforms that of EcoMapS up to 7% on average. The reason is that the ERTJA algorithm tries to use the working nodes and allocate the jobs of the failed node to them rather than an idle node. On the other hand, EcoMapS uses one existing idle node by waking it up at the beginning of the recovery phase in the cluster, and therefore, consumes additional energy.

Fig. 8 shows the behavior of the algorithms with respect to the probability of meeting deadlines after the node failure in Condition 1 of the Case 1 scenario. Since the EcoMapS just uses the sleeping node for handling the failure, and due to the fact that the failed not was overloaded,

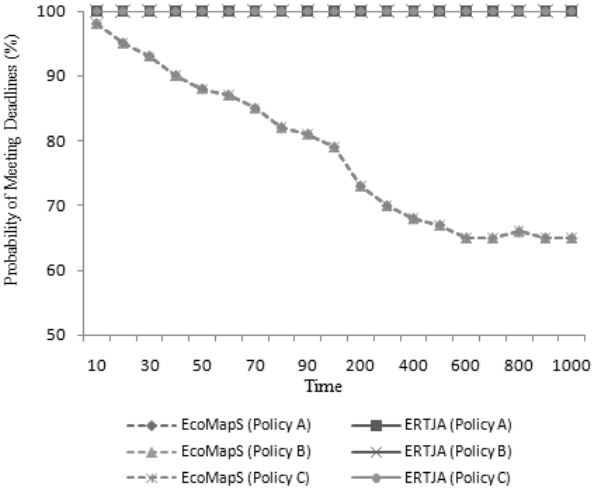

Fig. 8. Probability of meeting deadlines in EcoMapS and ERTJA (Condition 1 of Case 1)

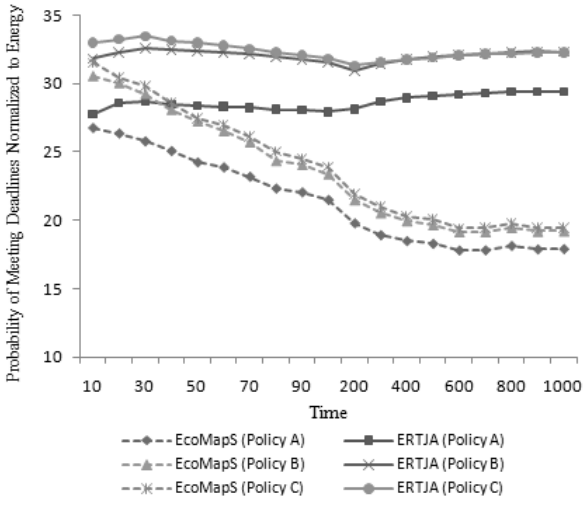

Fig. 9. Probability of meeting deadlines normalized to Energy in EcoMapS and ERTJA (Condition 1 of Case 1)

some job deadlines may be missed. However, as ERTJA exploits the idle times of less busy nodes too, besides turning on the sleeping node, all the jobs' deadlines are met. In these experiments, the behaviors of each algorithm for all policies (A, B, C) are similar, and therefore, the respective curves are overlapped. Normalized probabilities of meeting deadlines with respect to energy unit, which can be calculated as the ratio between the probability of meeting deadlines and the consumed energy, are also depicted in Fig. 9.

Figs.10 to 12 depict behavior of algorithms in Condition 2 of the Case1 scenario. In this condition it is assumed that there is no sleeping node to handle node failure in the cluster. Fig.10

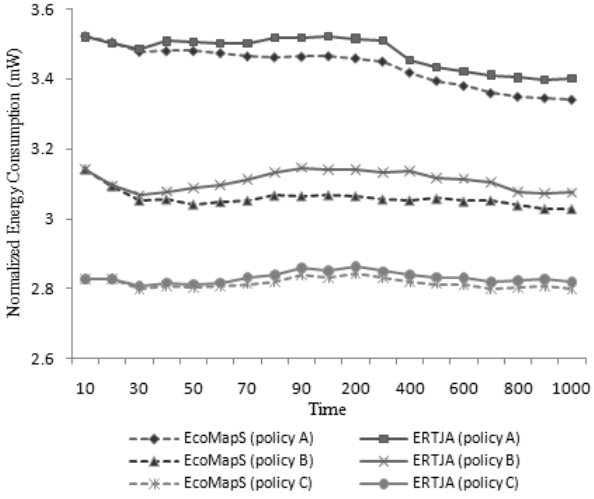

Fig. 10. Normalized energy consumption of EcoMapS and ERTJA (Condition 2 of Case 1)

shows the average energy consumption of the two indicated algorithms after node failure. As can be observed in Fig. 10, EcoMapS consumes less energy than ERTJA because it uses only one working node (with the maximum idle time) for handling the failure. Inversely, ERTJA can use all the working nodes to handle the node failure to guarantee the desired QoS of the cluster. According to Fig. 10, it can be comprehended that the energy conservation capabilities of EcoMapS compared to ERTJA is trivial. The probabilities of meeting deadlines for the same configuration are also depicted in Fig. 11. In policy C, the relatively low percentage of meeting deadlines is due to less free capacity of the active nodes. It can be observed that in this condition also the ERTJA can perform better than EcoMapS. Fig. 12 shows the Normalized probabilities of meeting deadlines with respect to energy units for the two algorithms. According to the figure, the ERTJA meet probability is larger than EcoMapS.

In a similar fashion, Figs. 13 to 18 illustrate the relative performance of the algorithms for

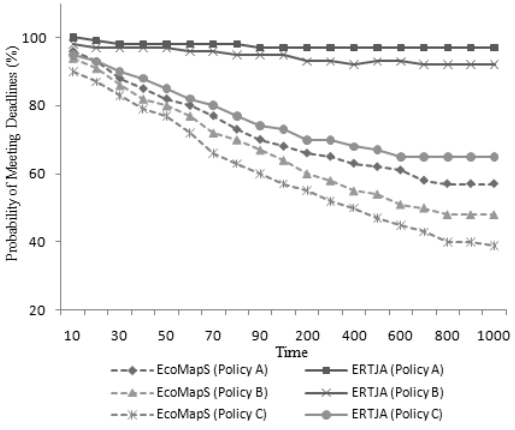

Fig. 11. Probability of meeting deadlines in EcoMapS and ERTJA (Condition 2 of Case 1)

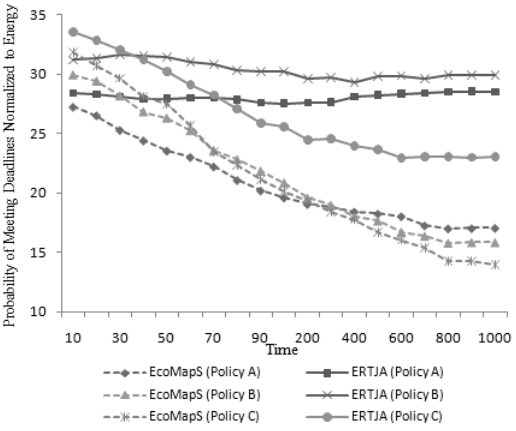

Fig. 12. Probability of meeting deadlines normalized to Energy in EcoMapS and ERTJA (Condition 2 of Case 1)

both Conditions 1 and 2 of the Case2 scenario. In the Case2 scenario, it has been assumed that an extremely overloaded node with a load of about 180% is corrupted. As can be observed, the relative behavior is similar to the Case 1 scenario.

According to our simulation results, ERTJA is more efficient than EcoMapS in the case of node failure. The reason is that the ERTJA algorithm conserves more energy than EcoMapS, and it can guarantee QoS of the cluster after node failure. Since the EcoMapS algorithm cannot

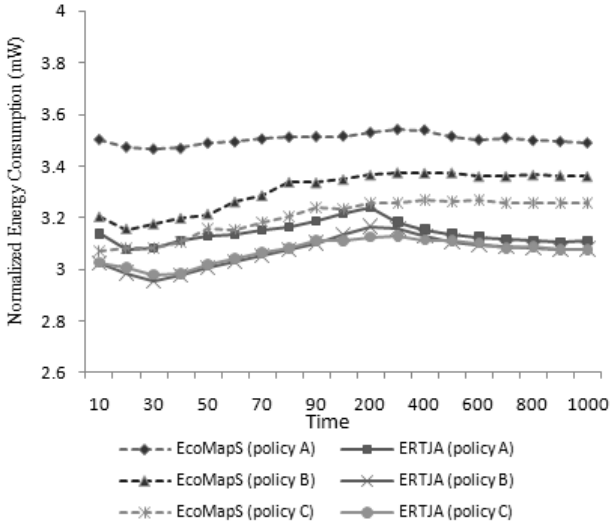

Fig. 13. Normalized energy consumption of EcoMapS and ERTJA (Condition 1 of Case 2)

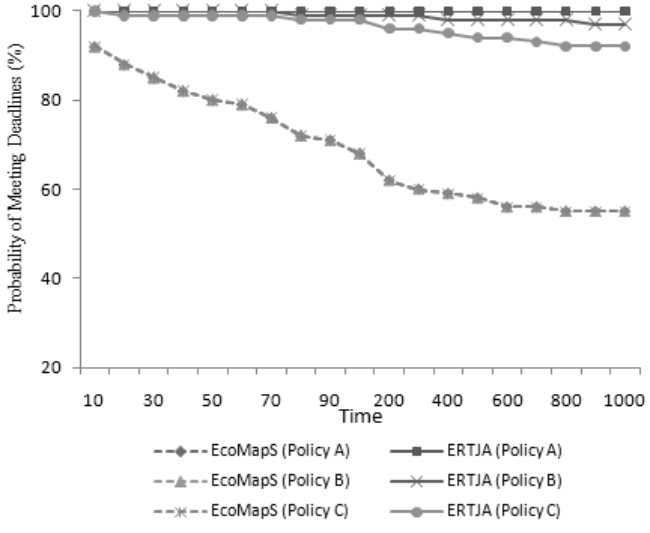

Fig. 14. Probability of meeting deadlines in EcoMapS and ERTJA (Condition 1 of Case 2)

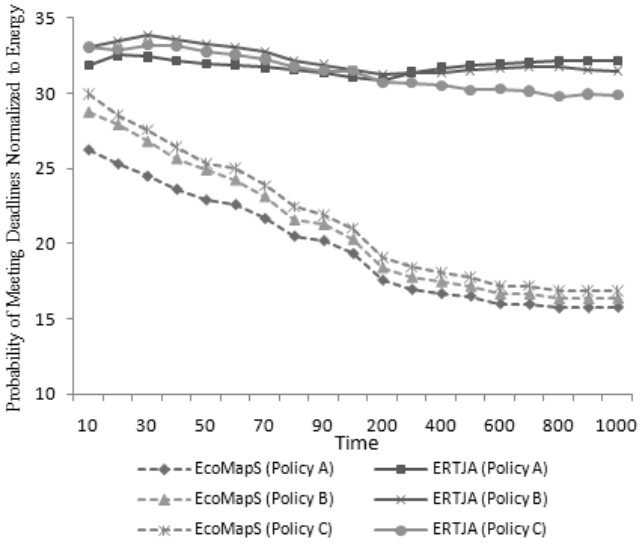

Fig. 15. Probability of meeting deadlines normalized to energy in EcoMapS and ERTJA (Condition 1 of Case 2)

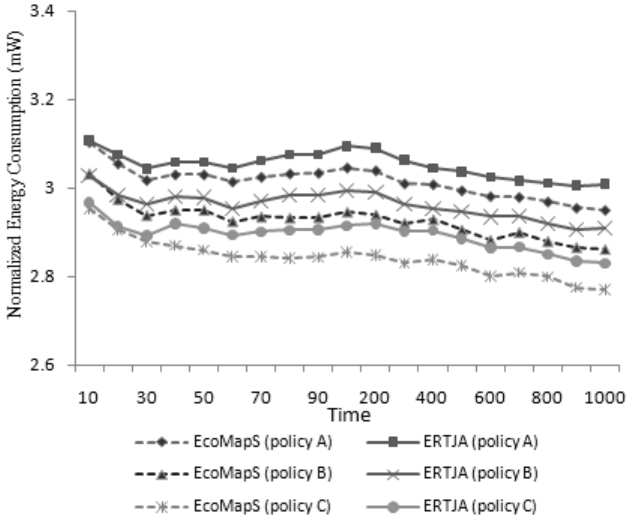

Fig. 16. Normalized energy consumption of EcoMapS and ERTJA (Condition 2 of Case 2)

guarantee deadlines of the failed node jobs, it is not applicable to real-time WSNs. On the other hand, ERTJA is a well-designed algorithm for such systems and it can guarantee the jobs' deadlines after node failure.

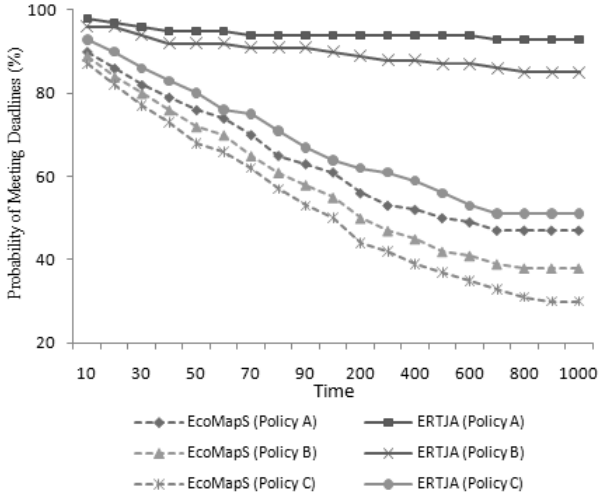

Fig. 17. Probability of meeting deadlines in EcoMapS and ERTJA (Condition 2 of Case 2)

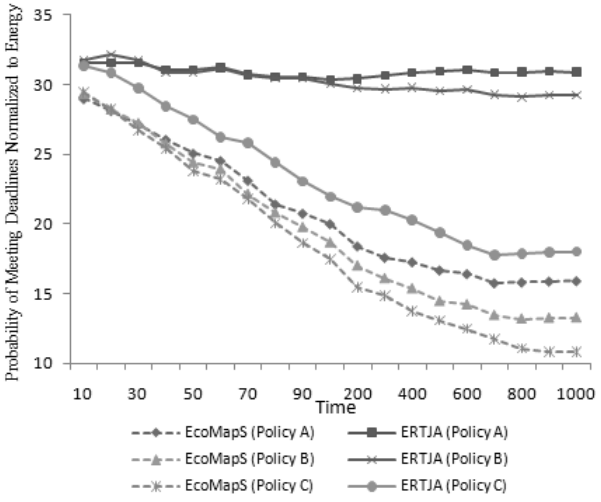

Fig. 18. Probability of meeting deadlines normalized to energy in EcoMapS and ERTJA (Condition 2 of Case 2)

7. CONCLUSION

In this paper, we have proposed a failure recovery algorithm for cluster-based WSNs, which tries to guarantee the cluster's QoS in an energy efficient manner upon the occurrences of node failures. ERTJA, which is a job allocation algorithm, is developed to allocate jobs of a failed node to the other active or sleeping nodes of the cluster. In order to do so, ERTJA uses a job admission control (LEJAC) for the allocation of jobs to the available nodes, and also exploits the resource augmentation technique. In fact, in the presence of enough idle time in the cluster

nodes, the proposed algorithm guarantees that the attained value by the cluster will certainly be the same as the failure-free cluster. Experimental results show that ERTJA provides superior performance in terms of energy consumption and probability of meeting deadlines with respect to the other studied algorithms. Furthermore, simulation results show that ERTJA accrues the maximum possible value in the case of limited available sleeping nodes in the cluster.

One idea for further studies is to extend this method to task sets with non-uniform values and also to adapt the proposed solution to heterogeneous WSNs.

REFERENCES

[1] W. Alsalih, S. Akl, and H. Hassancin, "Energy-aware task scheduling: towards enabling mobile computing over MANETs," in Proc.19th IEEE International Conference on Parallel and Distributed Processing, pp.51-59, 2005.

[2] M. Alghamdi, T. Xie, and X. Qin, "PARM: A Power-Aware Message Scheduling Algorithm for Real-Time Wireless Networks," in Proc. 1th ACM Workshop Wireless Multimedia Networking and Performance Modeling, pp.86-92, 2005.

[3] S. Baruah, G. Koren, B. Mishra, A. Raghunathan, L. Rosier, and D. Shasha, "On-line scheduling in the presence of overload," in Proc. 32th Annual Symposium on Foundations of Computer Science, pp.100-110, 1991.

[4] P. Basu, W. Ke, and T. Little, "Dynamic task-based anycasting in mobile ad hoc network," Mobile Network & Application, vol.8, pp.593-612, 2003.

[5] D. Estrin, L. Girod, G. Pottie, and M. Srivastava, "Instrumenting the world with wireless sensor networks," in Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing, vol.4, pp.2033-2036, 2001.

[6] S. Giannecchini, M. Caccamo, and C. Shih, "Collaborative Resource Allocation in Wireless Sensor Networks," Urbana, vol.51, pp.35-45, 2004.

[7] H. Jingcao and R. Marculescu, "Energy-aware communication and task scheduling for network-on-chip architectures under real-time constraints," in Proc. Europe Conference and Exhibition on Design, Automation and Test, pp.234-239, 2004.

[8] Z. Jinghua, L. Jianzhong, and G. Hong, "Tasks allocation for real-time applications in heterogeneous sensor networks for energy minimization," in Proc. 8th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, pp.20-25, 2007.

[9] B. Kalyanasundaram and K. Pruhs, "Speed is as powerful as clairvoyance," Journal of the ACM, vol.47, pp.617-643, 2000.

[10] C. Koo, T. Lam, T. Ngan and K. To, "Extra processors versus future information in optimal deadline scheduling," Theory of Computing Systems, vol.37, pp.323-341, 2004.

[11] R. Kumar, M. Wolenetz, B. Agarwalla, J. Shin, P. Hutto, A. Paul, and U. Ramachandran, "DFuse: a framework for distributed data fusion," in Proc. 1th international conference on Embedded networked sensor systems, pp.114-125, 2003.

[12] T. Lam and K. To, "Performance guarantee for online deadline scheduling in the presence of overload," in Proc. 12th Annual ACM-SIAM Symposium on Discrete Algorithms, pp.755-764, 2001.

[13] J. P. Lehoczky and S. Ramos-Thuel, "An optimal algorithm for scheduling soft-aperiodic tasks in fixed-priority preemptive systems," in Proc. Real-Time Systems Symposium, pp.110-123, 1992.

[14] J. Luo and N. K. Jha, "Power-Conscious Joint Scheduling of Periodic Task Graphs and Aperiodic Tasks in Distributed Real-Time Embedded Systems," in Proc. IEEE/ACM International Conference on Computer-aided Design, pp.357-364, 2000.

[15] C. Margi, R. Manduchi, and K. Obraczka, "Energy consumption tradeoffs in visual sensor networks," in Proc. 24th Brazilian Symposium on Computer Networks, 2006.

[16] C. Meesookho, S. Narayanan, and C. Raghavendra, "Collaborative classification applications in sensor networks," in Proc. 2th IEEE Workshop on Sensor Array and Multichannel Signal Processing,

pp.370-374, 2002.
[17] C. Phillips, C. Stein, E. Torng, and J. Wein, "Optimal time-critical scheduling via resource augmentation," Algorithmica, vol.32, pp.163-200, 2008.
[18] X. Qin, Z. Han, H. Jin, L. Pang, and S. Li, "Real-time fault-tolerant scheduling in heterogeneous distributed systems," in Proc. Workshop on Cluster Computing-Technologies, Environments and Application, 2000.
[19] S. Shivle, R. Castain, H. Siegel, A. Maciejewski, T. Banka, K. Chindam, S. Dussinger, P. Pichumani, P. Satyasekaran, and W. Saylor, "Static mapping of subtasks in a heterogeneous ad hoc grid environment," in Proc. 13th IEEE Workshop on Heterogeneous Computing, pp.21-32. 2004.
[20] J. Stankovic, K. Ramamritham, M. Spuri, and G. Buttazzo, "Deadline scheduling for real-time systems: EDF and related algorithms," Springer, 1998.
[21] S. R. Thuel and J. P. Lehoczky, "On-line scheduling of hard deadline aperiodic tasks in fixed-priority systems," in Proc. Real-Time Systems Symposium, pp.160-171, 1993.
[22] S. R. Thuel and J. P. Lehoczky, "Algorithms for scheduling hard aperiodic tasks in fixed-priority systems using slack stealing," in Proc. Real-Time Systems Symposium, pp.22-33, 1994.
[23] Y. Tian, E. Ekici, and F. Ozguner, "Cluster-based information processing in wireless sensor networks: an energy-aware approach," Wireless Communications and Mobile Computing, vol.7, pp.893-907, 2007.
[24] 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, pp.329-343, 2008.
[25] C. Yang, G. Deconinck, and W. Gui, "Fault-tolerant scheduling for real-time embedded control systems," Journal of Computer Science and Technology, vol.19, pp.191-202, 2004.
[26] Y. Yu and V. Prasanna, "Energy-balanced task allocation for collaborative processing in wireless sensor networks," Mobile Networks and Applications, vol.10, pp.115-131, 2005.

Hamid Karimi
received his B.S. degree in Computer Engineering from the University of Applied Science and Technology, Mashhad, Iran, in 2006. He recently received his M.S. degree in Information Technology from the School of Electrical and Computer Engineering at the University of Tehran, Iran, in 2010. He currently is a researcher in the Router Laboratory, at the University of Tehran. His research interests include wireless sensor networks, real-time and distributed systems with a focus on ubiquitous, pervasive, sensor and networked computing.

Mehdi Kargahi
IEEE Member is currently an assistant professor in the School of Electrical and Computer Engineering at the University of Tehran, Iran. He received his B.S. degree in Computer Engineering from Amir-Kabir University of Technology (Tehran Poly-Techniques) in Tehran (1998) and both his M.S. (2001) and Ph.D. (2006) in Computer Engineering from Sharif University of Technology in Tehran. He visited the University of Victoria in Canada as a researcher in 2006 and has also been a researcher in the School of Computer Science at the Institute for studies in theoretical Physics and Mathematics (IPM) from 2003. His research interests include performance modeling and power management of dependable and distributed real-time systems with stochastic properties.

Naser Yazdani

is an associated professor in the School of Electrical and Computer Engineering in the University of Tehran currently. He got his B.S. in Computer Engineering from Sharif University of Technology, Tehran, Iran. He worked in Iran Telecommunication Research Center (ITRC) as a consultant, researcher and developer for a few years. To pursue his education, he entered Case Western Reserve University, Cleveland, Ohio, USA, later and graduated with a PhD in computer science and Engineering. Then, Dr. Yazdani worked in different companies and research institutes in USA. He joined the ECE Dept. of Univ. of Tehran, Tehran, Iran, at Sep. 2000. Then, Dr. Yazdani initiated different research projects and labs in high speed networking and systems. His research interests include Networking, packet switching, access methods, Operating Systems and Database Systems.