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

, Mehdi Kargahi, Nasser Yazdani
School of Electrical and Computer Engineering
College of Engineering
University of Tehran, Iran
{, kargahi, yazdani}@ut.ac.ir

***Abstract*— Guaranteeing quality of real-time services in wireless sensor networks (WSN) requires efficient resource management, especially for energy resources. This is due to the fact that nodes in such networks usually use limited non-replaceable batteries. On the other hand, the nodes of WSNs often work in harsh environments, and therefore, susceptible to failure occurrences due to environmental affects or exhaustion of their battery. In this paper, we present a dynamic energy efficient real-time job allocation algorithm called ERTJA for handling node failures in a cluster. ERTJA tries to minimize the energy consumption of the cluster by minimum activation of sleeping nodes while guaranteeing the QoS of the application assigned to the cluster at the same previous level. Further, when the number of sleeping nodes is limited, the proposed algorithm uses the idle times of active nodes to have a graceful QoS degradation of the cluster upon node failure. Simulation results show significant performance improvements of ERTJA in terms of energy consumption comparing to the N-EDF-Plus algorithm. According to the results, ERTJA can save up to 26.5% of the cluster's energy consumption with respect to N-EDF-Plus in the studied scenarios.**

Keywords- Wireless sensor network, Real-time scheduling, Job allocation, Failure recovery, Quality of service.

I. INTRODUCTION

Real-time networked and distributed embedded systems such as wireless sensor networks (WSN) have increasingly became popular since the last decade [4], [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 allocated to available sensor nodes of a cluster to fulfill a common real-time mission [23], [1]. As another example of challenging applications in this area, we can mention to video sensor networks. Since multimedia processing generally involves computation-intensive operations, a single sensor node may not be able to complete the respective jobs [7].

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 or quite costly that it will overcome any benefits drawn from the WSN. Consequently, most of the research around WSN focuses predominately on energy saving problems. One typical mission-critical real-time sensor network absolutely has the following two characteristics: one is guaranteeing the desired timing constraints and second is continuing the specified mission in spite of hardware or software failure. For example, in embedded control systems (ECS) 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 ECS is to complete all real-time tasks within their specified timing constrains even in the presence of faults [24]. Accordingly, fault-tolerance (and/or fault-recovery) techniques as well as real-time resource management (or scheduling) algorithms should be used together 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. 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 allocated and scheduled in an offline manner among 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. This is done through a task reallocation algorithm, which dynamically assigns the jobs of the failed tasks (previously located on the failed node) to the remaining failure-free nodes in the cluster. This algorithm tries to avoid excessive turning on of the sleeping nodes to save as 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 in the same acceptable level even after an overloaded node failure,
- It tries to heuristically turn on the minimum number of sleeping nodes in the cluster to save more energy,
- The algorithm tries to accrue the maximum possible QoS in the cases which, after failure, there is not enough sleeping nodes in the cluster to be turned on to satisfy the target QoS.

The rest of this paper is organized as follows. In the next section we discuss the related work. Section III describes the system, task and energy consumption models. In Section IV, we describe a resource augmentation method which is used as a basic idea in our algorithm. Section V proposes our job allocation algorithm, and presents an online job admission control in details. Section VI discusses experimental results. Finally, we conclude the paper in Section VII.

978-1-4244-4996-5/09/$25.00 ©2009 IEEE

II. Related Work

Task allocation has been well studied in the area of distributed system [23],[25]; however, in many studies in the literature of scheduling in real-time WSNs, the effects of fault occurrences in the systems are ignored, whereas they have big impacts on the performance and QoS of the systems.

EAS [6] statically schedule communication and computation onto heterogeneous Network-on-Chip architectures under real time constraint. They focus on the architectures interconnected by 2D mesh networks with XY routing schemes. CoRAl [5] is an online task scheduling mechanism proposed to allocate network resources to tasks of periodic applications in WSN. 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 [3]. DFuse [10] is a data fusion task mapping mechanism for WSN. None of these solutions tolerate nodes failures. Task mapping and scheduling heuristics are presented in [18] for heterogeneous mobile ad hoc grid environments. EcoMapS [22] 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 [17], 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 allocate main and backup copies of tasks to sensor nodes that leads to an increase in the reliability as maximum as possible. The cost of this manner is high and causes much overhead to the system.

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

III. System, Application and Energy Consumption Model

A. 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. Each cluster executes an application which is either assigned during the network setup time or remotely distributed by base stations during the network operation. Once assigned, applications are independently executed within each cluster unless either one nodes is corrupted or finished its energy. The intrinsic tasks of application are assigned to the nodes of a cluster according to an off-line schedule. Some of these nodes may be scheduled in an overload condition. Without loss of generality, we apply the remaining of our discussions within a single cluster.

B. 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 as 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* of each task. 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.

C. 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: **Sleep** state consumes the lowest power. The processor core is in sleep mode and all hardware components are deactivated. Power value (joule 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. The 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, power value is ψ_K. To formulating energy consumption; we assume that $\boldsymbol{t}$ is the time duration that a sensor node is involved in the system. $K(t)$ is total time which sensor nodes are active. S(t) and I(t) also have similar definition for sleep and idle mode, then we have :

$$k(t) = \sum_{i=1}^{n} \sum_{t=1}^{t} K_{it} \qquad (1)$$

$$I(t) = \sum_{i=1}^{n} \sum_{t=1}^{t} I_{it} \qquad (2)$$

$$S(t) = \sum_{i=1}^{n} \sum_{t=1}^{t} S_{it} \qquad (3)$$

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

$$E_{total} = E_K + E_S + E_I, \qquad (4)$$

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

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), initial schedule of the node is violated. In this situation, jobs of the failed node miss their deadlines. Thus a recovery procedure is needed to avoid of QoS degradation in the cluster. Recovery procedure should be dynamically allocates the jobs of the failed tasks (previously located on the failed node) to the remaining failure-free nodes in the cluster.

Since initial schedule (offline schedule) is violated, jobs of the failed node are unknown to the recovery procedure before their release times. Thus, an online scheduling algorithm should be used to handle failure, and to execute jobs in the other available nodes. Online scheduling is not an optimal algorithm in overload condition while offline scheduling is optimal [2]. To overcome this problem and guarantee system Qos after the node failure, we exploit resource augmentation technique to have an optimal on-line scheduling [9], [11], [8], [16]. This technique is described in the next section.

IV. 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 [19] for a survey). From a theoretical viewpoint, EDF is optimal for scheduling underloaded systems, i.e., whenever there exists a schedule meeting the deadlines of all jobs released, EDF can always do so [19]. However, when the system is possibly overloaded, no algorithm has a worst-case performance guarantee in the sense that the performance cannot match or be competitive against the offline adversary [2]. In recent years, a plausible approach to studying better performance guarantee for online scheduling is to allow the online scheduler to use more processors than the offline adversary. Use of more processor in online scheduling is to compensate the online scheduler for the lack of future information. Literature [9] proposed an online scheduling algorithm called N-EDF-Plus which is a 3-processor optimal for online scheduling jobs. The paper proves that to schedule any task set T with uniform *value density* (tasks have a value proportional to its computation time), 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.

V. PROPOSED ALLOCATION ALGORITHM

Guaranteeing quality of services (QoS) in sensor networks depends critically on tolerance to 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 the case of node's 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 improve the energy saving of the cluster in case of failure. To make the best trade-off between QoS-guarantee and energy saving, we employ a novel job admission control algorithm which executes in the cluster head and chooses a sensor node for the execution of the failed node jobs. Details of this algorithm described in the following section.

A. Algorithm Description

ERTJA exploits the N-EDF-Plus algorithm proposed in [9] to guaranteeing QoS and getting the optimum value from jobs of the tasks of a failed node. ERTJA is a modified version of the N-EDF-Plus which follows the steps of the latter algorithm in a virtual manner. The N-EDF-Plus algorithm is a 3-processor optimal algorithm. The algorithm is used for online scheduling of jobs with uniform value densities with 3 empty (sleeping) nodes for getting values identical to an off-line optimal schedule. Requesting 3 sleeping nodes from system is costly, and is impossible in most of the times due to resource constraints in such systems. Thus, this algorithm may not be used directly in sensor networks. On the other hand, ERTJA virtually provides 3 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 case of node failure. When a node failure is occurred, 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 by minimum activation of sleeping nodes. On the other hand, to save energy, nodes should remain sleeping as much as possible.
In case of a node failure, ERTJA begins with 3 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 regards, once S_e admits a job, the job is guaranteed to be completed by its deadline. After admission of J by S_e, ERTJA calls Online Job Admission Control (OJAC) algorithm for mapping the job to an available node (working or sleeping) in the cluster. (OJAC algorithm which is used to replace a virtual node with a real working or sleeping node is described in details in the next subsection). 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 OJAC 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, OJAC maps the job on a real node. The ERTJA algorithm is described in detail in Fig. 1.

B. Online Job Admission Control (OJAC)

In this section, we present a novel admission control scheme for a mixed set of periodic tasks and a number of new jobs with hard deadlines, which can achieve near optimal performance. 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 OJAC, is based upon the slack stealing method, introduced by Thuel and Lehoczky [13],[21],[20]. Using slack stealing, OJAC provides a method for guaranteeing and scheduling new jobs with periodic tasks in the sensor nodes. With complete knowledge of the node's execution and future periodic requirements, OJAC can determine whether there is an adequate time in the node to admit a new job. For the reallocation of jobs of the failed node, OJAC is executed in the cluster-head, and gathers information of the periodic tasks in the working nodes of the cluster. The OJAC, 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.

```
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 OJAC (J)
    else if Sd is idling
        Sd runs J
        Call OJAC (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 OJAC (J)
When Sd completes a job:
    if P ≠ ∅
        Let J be the job in P with the latest deadline
        P ← P − {J}  // if some Su is running J , it becomes idle
        Sd runs J
        Call OJAC (J)
When J ∈ P becomes urgent (slack time become zero):
    if some Su is idling
        Su runs J
        Call OJAC (J)
    else
        Let J' be the urgent job other than J in P with earliest deadline
        Let Su be the processor running J'
    if d(J) > d(J')
        P ← P − {J'};
        Su runs J;
        Call OJAC (J)
    else
        P ← P − {J}
```

Fig. 1.The ERTJA algorithm

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 DWP method, each task's execution is postponed toward its deadline as long as does not miss its deadline [12].

Using the above mentioned tables as well as slack information, OJAC determines whether there is enough slack to support the extra required resource in a working node. If there is adequate slack, then the job is mapped to the node. Otherwise, the job is checked for admission in another node. After assigning a job to one of the working nodes, the T-EDF and T-DWP tables of the node are updated. Therefore, OJAC is used to ensure that (1) only those jobs that will complete by their deadline are admitted, and (2) a job that is assigned to the node can only consume the slack in the working node and does not interfere with any previously task on the same node.

Example. Suppose that a node has a periodic task set with two tasks, T_1 and T_2, having the computation requirements, C_1=1 and C_2=2, and the periods, P_1=4 and P_2=6, respectively. These tasks are to be pre-assigned over a single hyperperiod, 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= "122102210000"
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 T_1, and an time unit consumed by T_2. 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 OJAC algorithm is:

$$S(D_j) = D_j - R_j - (CR - CP), \qquad (5)$$

where R_j, D_j denote release time, deadline of the new job. CP and CR respectively denote all the computation processing done until release time of the new job (the current scheduling time) and the entire computation requirement until the deadline of the new job. In this regards, the job is admitted when $S(D_j) \geq C_j$ (C_j is computation requirement of job J) . The details of OJAC are depicted in Fig. 2.

Suppose that job J is released at instant 5 (R_j=5), its deadline is 8 (D_j=8) and its computation time is 2 (C_j=2). To allocate job J to one of the working nodes, the OJAC algorithm calculates computation requirement (CR) from instant 1 to 8 in T-DWP table (*CR*=3). Then, it calculates computation processing done from instant 1 to 5 in T-EDF table (*CP*=4). Consider the above calculated T-EDF and T-DWP tables. We have:

$$S(8) = 8 - 5 - (3 - 4).$$

Therefore, since $S(8) \geq C_j$, the job is admitted.

```
Initialization: flag_admission ← false
Foreach Si in set of working nodes
    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 -(CR – CP)
    // S(DJ) is number of slack until time DJ
    If S(DJ) ≥ CJ   // CJ is Computation requirement J
        Si admit J;
        Update T-EDF & T-DWP of Si
        flag_admission=true; break;
If not (flag_admission)
    Sk in set of sleeping nodes admit J
```

Fig. 2. The OJAC algorithm

VI. SIMULATION RESULT

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. Through our simulations we investigate the efficiency of the algorithms with respect to average energy consumption and average number of sleeping nodes which are activated to handle possible failures. We assumed that there are 10 homogeneous sensor nodes in a single-hop cluster. The initial energy of each node is considered as 5000 mJ. The power consumption of each node is shown in Table I. Hence, all result provided in this section are averaged over one hundred experiments per endpoint.

A task set consists of 20 independent tasks with relative deadlines equal to their respective periods is considered in our simulations. 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 are used to generate the task period time and the task execution (service) time. The task's period time are uniformly distributed between [10,50] while task's 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 exceed no further than the acceptable utilization of the policy. These utilizations are listed in Table II. Typically, these policies are used in the various WSNs for the sake of load balancing and lifetime prolonging. The tested scenario considers failure occurring in an overloaded node that exists in the cluster. (It is possible that some nodes be overloaded due to the deficiency of the allocation algorithm or system defect).

In Case 1, we assume that after running each pre-allocation policy A, B and C, an overloaded node with utilization 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 underloaded nodes. In each of tests (Case 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 to its steady state behavior, when it has passed a 1000-second time period. We consider that the failure has been occurred in the most damaging time during a hyperperiod, i.e., as 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 (average energy consumption produced by one hundred experiments).

TABLE I
POWER CONSUMPTION OF WIRELESS SENSOR NETWORK

Mode of Sensor	Ratio
Sleep	0.001
Idle	0.8
Active	1

TABLE II
ACCEPTABLE UTILIZATION OF POLICIES

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

The N-EDF-Plus is an optimal on-line scheduling algorithm especially for the overloaded nodes. However, it consumes unnecessary energy because it does not use of the available capacity of any working nodes of the cluster. On the other hand, the ERTJA algorithm uses the working (underloaded) nodes, and tries to allocate the jobs of the failed node to them. As can be observed in Fig. 3, energy saving of ERTJA outperforms that of N-EDF-Plus up to 26.5% averagely.

In Case 2, to observe the system behavior in the case of failure of an extremely overloaded node, the node utilization is considered about 180%. As shown in Fig. 4, ERTJA can save up to 22.9% of energy consumption averagely comparing to N-EDF-Plus.

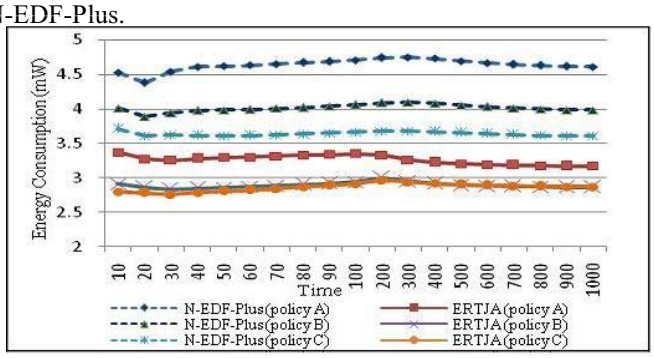

Fig. 3. Energy consumption of N-EDF-Plus and ERTJA (Case 1)

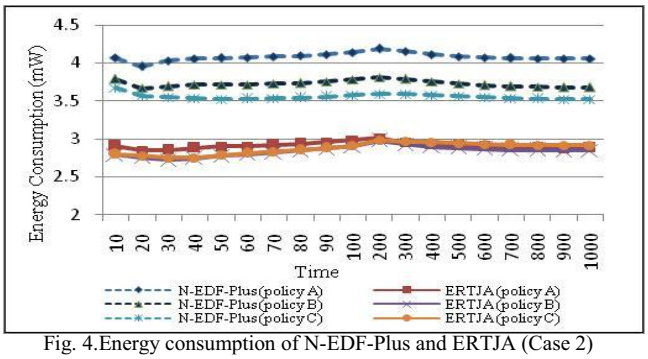

Fig. 4.Energy consumption of N-EDF-Plus and ERTJA (Case 2)

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 average number of added nodes (before and after node failure). According to Fig. 5 and 6, the N-EDF-Plus algorithm needs 2 or 3 sleeping nodes for guaranteeing the

cluster's QoS and failure handling while ERTJA algorithm needs only 1 sleeping node averagely. Since in many situations, there may not exist enough sleeping nodes to awakened, ERTJA can have considerable preference with respect to 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.

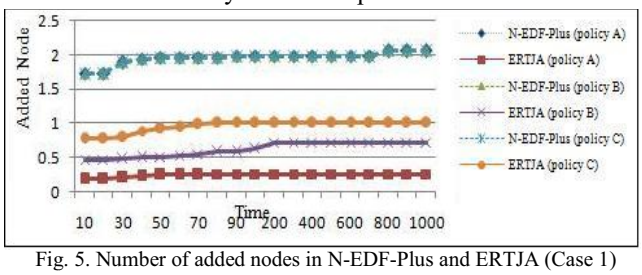

Fig. 5. Number of added nodes in N-EDF-Plus and ERTJA (Case 1)

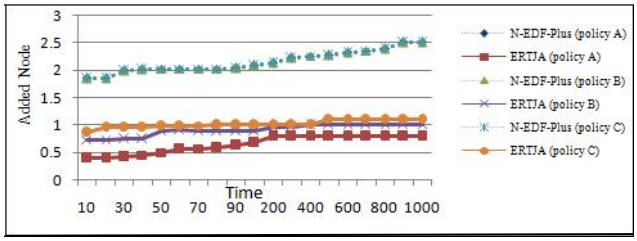

Fig. 6. Number of added nodes in N-EDF-Plus and ERTJA (Case 2)

VII. CONCLUSION

In this paper, we address the issue of allocating jobs of a failed node in a cluster of WSNs to the other active or sleep nodes, with the objective of guaranteeing quality of real-time services. The presented real-time job allocation algorithm, ERTJA, handles the failure of the node in a single-hop clustered WSN. In case of node failure, ERTJA tries to minimize energy consumption while guaranteeing the same previous level of QoS of the application assigned to the cluster. The proposed solution guarantees that the attained value by the cluster nodes will certainly be the same or even better than the conditions of the cluster before the failure. ERTJA uses a novel job admission control for the allocation of jobs to the available nodes, and also exploits the resource augmentation technique to guaranteeing the QoS. Experimental results show that ERTJA provides superior performance in terms of energy consumption with respect to the N-EDF-Plus algorithm. Furthermore, simulation results show that ERTJA accrue the maximum possible value in the case of limited available sleep nodes in the cluster. Our future work includes extending this method to task sets with non-uniform value densities and also adapting 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 conference on Parallel and Distributed Processing Symposium. 2005.

[2] 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 Computer Science.1991.pp 100-110.

[3] P. Basu, W. Ke, and T. Little, "Dynamic task-based anycasting in ad hoc networks". Mobile Networks and Applications. 8 (5). pp. 593-612. 2003.

[4] D. Estrin, L. Girod, G. Pottie, and M. Srivastava, "Instrumenting the World with Wireless Sensor Networks". in *Proc.* IEEE international conference on Acoustics, Speech, Signal Processing. 2001. pp. 2033-2036.

[5] S. Giannecchini, M. Caccamo, and S. Chi-Sheng, "Collaborative resource allocation in wireless sensor networks". in *Proc.* 16th Euromicro Conference on Real-Time Systems. 2004. pp. 35-44.

[6] 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. 2004. pp. 234-239.

[7] Z. Jinghua, L. Jianzhong, and G. Hong, "Tasks Allocation for Real-Time Applications in Heterogeneous Sensor Networks for Energy Minimization". in *Proc.* Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing. 2007. pp. 20-25.

[8] B. Kalyanasundaram and K. Pruhs, "Speed is as powerful as clairvoyance". Journal of the ACM. 47(4). pp. 617-643. 2000.

[9] C. Koo, T. Lam, T. Ngan, and K. To, "Extra processors versus future information in optimal deadline scheduling". Theory of Computing Systems, 37(3). pp. 323-341. 2004.

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

[11] 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. 2001.pp. 755-764.

[12] J. Lee, S. Lee, and H. Kim, "Scheduling soft aperiodic tasks in adaptable fixed-priority systems". ACM SIGOPS Operating Systems Review. 30 (4). pp. 17-28. 1996.

[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. 1992. pp. 110-123.

[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. 2000. pp. 357 – 364.

[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. Phillips, C. Stein, E. Torng and J. Wein, "Optimal time-critical scheduling via resource augmentation". Algorithmica. 32(2). pp.163-200. 2008.

[17] X. Qin, Z. Han, H. Jin, L. Pang, and S. Li, "Real-time fault-tolerant scheduling in heterogeneous distributed systems". in *Proc.* International Workshop on Computing-Technologies, Environments, Application. 2000.

[18] 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.* Parallel and Distributed Processing Symposium. 2004.

[19] J. Stankovic, "Deadline scheduling for real-time systems: EDF and related algorithms". Springer. 1998.

[20] 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. 1993. pp. 160-171.

[21] 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. 1994. pp. 22-33.

[22] Y. Tian, E. Ekici, and F. Ozguner, "Energy-constrained task mapping and scheduling in wireless sensor networks". in *Proc.* Workshop on Resource Provisioning and Management in Sensor Networks. 2005.

[23] T. Xie and X. Qin, "An Energy-Delay Tunable Task Allocation Strategy for Collaborative Applications in Networked Embedded Systems". IEEE Transactions on Computers, 57 (3). pp. 329 – 343. 2008.

[24] C. Yang, G. Deconinck, and W. Gui, "Fault-tolerant scheduling for real-time embedded control systems". Journal of Computer Science and Technology.19(2). pp. 191-202. 2004.

[25] Y. Yu and V. Prasanna, "Energy-balanced task allocation for collaborative processing in wireless sensor networks". Mobile Networks and Applications. 10 (1). pp. 115-131. 2005.