Hindawi
Wireless Communications and Mobile Computing
Volume 2017, Article ID 8382026, 6 pages
https://doi.org/10.1155/2017/8382026

Research Article

E-Token Energy-Aware Proportionate Sharing Scheduling Algorithm for Multiprocessor Systems

Pasupuleti Ramesh[1] and Uppu Ramachandraiah[2]

[1]*Department of Electronics and Communication Engineering, Hindustan University, Chennai, India*
[2]*Department of Electronics and Instrumentation Engineering, Hindustan University, Chennai, India*

Correspondence should be addressed to Uppu Ramachandraiah; uppur@hindustanuniv.ac.in

Received 2 July 2016; Revised 22 December 2016; Accepted 11 January 2017; Published 15 February 2017

Academic Editor: Xianfu Lei

WSN plays vital role from small range healthcare surveillance systems to largescale environmental monitoring. Its design for energy constrained applications is a challenging issue. Sensors in WSNs are projected to run separately for longer periods. It is of excessive cost to substitute exhausted batteries which is not even possible in antagonistic situations. Multiprocessors are used in WSNs for high performance scientific computing, where each processor is assigned the same or different workload. When the computational demands of the system increase then the energy efficient approaches play an important role to increase system lifetime. Energy efficiency is commonly carried out by using proportionate fair scheduler. This introduces abnormal overloading effect. In order to overcome the existing problems E-token Energy-Aware Proportionate Sharing (EEAPS) scheduling is proposed here. The power consumption for each thread/task is calculated and the tasks are allotted to the multiple processors through the auctioning mechanism. The algorithm is simulated by using the real-time simulator (RTSIM) and the results are tested.

1. Introduction

Wireless sensor networks (WSNs) are used for healthcare systems, Supervisory Control and Data Acquisition (SCADA) systems, smart grids, agriculture and environment monitoring, public safety, and military systems. Sensors in WSNs are predictable to run unconventionally for a longer period of time in order to increase the network lifetime. It is not possible always to manually reload the jots due to the conservation cost and unapproachability. In WSN's multicore or multiprocessor architectures emerging advancements are focused on making them more effective in processing different tasks without any deadline missing. These architectures may be homogeneous or heterogeneous and must be effective in the aspects like size, processing speed, power consumption, and so forth. The response of the real-time system for different incoming resources at different time intervals is the major obligation for an efficient and effective system. Generally, in multiprocessor architecture scheduling is decided based on the system applications, types of inputs, type of the processor and response of the system, and so on. Based on these parameters a new scheduling algorithm is proposed here. It is flexible and compatible with different tasks. In this scheduling the energy efficiency is achieved through the task allocation based on its energy consumption. As per the sorting mechanism the task allocation is done with advantage of less energy consumption compared to the other techniques.

This paper is ordered as follows: in the subsequent section, we presented the contemporary prevailing scheduling schemes. In Section 3, we recommend EEAPS algorithm. In Section 4, we originate the hardware details of multiprocessor systems and Section 5 deliberates real-time simulator model. In Section 6, we analyze the EEAPS power and energy graphs for different threads. Lastly, Section 7 concludes this paper.

2. Related Works

Anastasi et al. present energy-preservation structures in wireless sensor networks [1]. Here the writers concentrated on duty cycle and data-saving methods. Contemporary reviews reveal that numerous methods are projected for energy effectiveness like Energy-Aware routing protocols [2], data

(1) for $i = 1$ to N do
(2) for $k = 1$ to M do
(3) Calculate Power consumption of the task

$$P_{i,k} = \left[(\beta_k - \gamma_k) * \left(\frac{M}{\alpha_i} \right) \right] + \gamma_k$$

(4) Calculate Energy consumption of the task

$$E_{\tau_i} = P_{i,k} * C_i$$

(5) if $(E_{\tau_i} < E_{\text{th}})$ then
(6) Call Processor1()
(7) else if $(E_{\tau_i} == E_{\text{th}})$ then
(8) Call Processor2()
(9) else Call Processor3()
(10) end if
(11) end if
(12) Calculate total energy consumption of the clients

$$E_{\text{clients}} = \sum_{i=1}^{N} \sum_{k=1}^{M} P_{i,k} * C_i$$

(13) end for
(14) end for
(15) Calculate Total Energy consumption of the System

$$E_{\text{Total}} = E_{\text{server}} + \sum_{i=1}^{N} \sum_{k=1}^{M} P_{i,k} * C_i$$

Algorithm 1: Sample EEAPS algorithm.

accumulation systems [3, 4], and energy gathering tactics [5]. A scheduling algorithm combining the energy efficiency and the performance for proportional-share scheduling of threads in a heterogeneous processor is developed in [6]. An optimal scheduling algorithm for uniprocessors is real-time static voltage and frequency scaling is projected in [7]. Tasks models with both fixed and migrating tasks are considered with the existence of fair scheduling proved in [8]. Both the tasks are scheduled differently between the resources at several times. A periodic schedule is produced by the fair scheduler. Here, each task is allocated to a time which is the same as its period [9]. Through a static analysis and runtime scheduling, the proposed scheduling maps the program to the best fit processor which is designed based on the program phases in [10]. Here it is showed that the energy efficiency for the phase based scheduling is achieved by comparing this against the statical mapping and the periodic sampling. To synthesize multiprocessor implementations of hard real-time system with independent periodic tasks many algorithms are presented in [11]. Goossens et al. present Pfair scheduler which is optimal for periodic tasks on multiprocessor systems [12]. In Pfair tasks are obviously required to make evolution at steady rates. Baruah et al. proposed two optimal Pfair schedulers named PF [13] and PD [12]. Both algorithms assign priority based on deadline. If two subtasks have the same deadline then PF compares future subtask deadlines, which is expensive technique but PD inspects four tie-break parameters. A new proof is demonstrated for the Pfair scheduling algorithm which is optimal for the sporadic tasks in multiprocessor system [14]. A hierarchical Pfair algorithm is proposed with allocation constraints in [15]. Here the amount of time by which the fixed task is missed is bounded. A task dependent job allocation for a multiprocessor system is shown in [16]. Energy characteristics analysis of parallel algorithms for scalable multiprocessor processors is evaluated by a methodology in [13] while satisfying the performance requirements. It is quite different in scalability for parallel algorithm with power and performance. Using this method, we can determine number of processors to be used to minimize energy consumption. Anderson and Srinivasan presented ER-fair and offered new algorithm called PD^2. It is also called mixed Pfair or modified Pfair (Pfair-M). It is optimal for both early release and nonearly release periodic tasks [17]. Based on literature survey we understood that many of the scheduling algorithms suffer from NP-hard problem due to nonpreemption mechanism and minimization of total completion time. The two rules earliest completion time and earliest release time are heuristic algorithms, and for them accuracy bounds are unknown.

3. The Proposed Work

Here, we proposed an energy efficient scheduling algorithm based on preemptive mechanism called E-token Energy-Aware Proportionate Sharing (EEAPS) scheduling algorithm. For multiprocessor architecture based on homogeneous structure the processors are classified with respect to threshold energy. The task set is split into number of threads/tasks and their energy requirement is calculated. Thus to perform the task allocation among the processors based on their energy requirement, this is called auctioning mechanism. The processor power consumption is calculated prior to the tasks to decide the task allocation after their arrival. The incoming tasks/threads are analyzed with their characteristics for the application in the system. Sample EEAPS algorithm is represented in the Algorithm 1.

3.1. Task Model. The system consists of a set of n independent and m dependent tasks $\Gamma = \{\tau_1, \tau_2, \ldots, \tau_n\}$.

The characteristics of each task τ_i are represented as (C_i, T_i, d_i) and here C_i denotes task worst case execution time, T_i is the time period, and d_i is the dependency of the task. The energy of each task E_{τ_i} is calculated and compared with the threshold energy E_{th} of the system. Task (τ_i) allocation to the processors is defined as the function of energy and time. It is represented as

$$F_{\tau_i}(E,t) = \begin{cases} \text{Processor1} & E_{\tau_i} < E_{\text{th}} \\ \text{Processor2} & E_{\tau_i} = E_{\text{th}} \\ \text{Processor3} & E_{\tau_i} > E_{\text{th}} \\ \text{Processor4} & \text{for Sporadic Tasks.} \end{cases} \quad (1)$$

Here E_{th} is the average energy of the available processors. Periodic tasks follow fixed priority mechanism based on their periods and here task deadline $D_i = T_i$ and it allows preemption mechanism whenever high priority task arrives. Aperiodic tasks follow dynamic priority. Sporadic tasks are directly allotted to processor4 without checking the energy requirement. If two or more sporadic tasks arrive simultaneously, then priority is assigned to the sporadic tasks depending on its deadline.

The total utilization of the task set U_i is always within the available processor capacity (c) of the system.

$$U_i = \sum_{i=1}^{n+m} \left(\frac{C_i}{T_i}\right) \leq c \quad (2)$$

(see [18]). For the defined task set, EEAPS is feasible because it satisfies the following condition:

$$\left[\frac{C_1}{T_1}\right] + \left[\frac{C_2}{T_2}\right] + \cdots + \left[\frac{C_N}{T_N}\right] \leq 1 \quad (3)$$

(see [18]). Here $T_1, T_2, \ldots, T_N$ are time period of the tasks and $C_1, C_2, \ldots, C_N$ are task's execution times. In order to minimize the complexity (NP-hard) of the algorithm, this work proposed competent heuristic approach. It is energy efficient and also compatible for different tasks set. This model adopts global approach for multiprocessor task allocation and scheduling based on task's energy requirement.

3.2. Processor Model. The system consists of P homogeneous k processors $P_1, P_2, \ldots, P_k$. That means all processors have equal voltage and frequency ratings and exhibit similar computational capacities. Here we considered ARM7TDMI LPC2148 microcontrollers for implementation. It is compatible with RT-Linux and μc-Linux. Its specifications are given in Table 1.

3.3. Energy Model. Energy (E) consumed by the processor which can be represented in terms of the power consumption (P) is

$$E = P * t. \quad (4)$$

Table 1: ARM7TDMI specifications.

S. number	Parameters	Specification
1	Operating voltage	3.3 V
2	Current	100 mA–500 mA
3	ADC	16 channels with 10-bit resolution
4	UARTs	2

Here "t" is the processor task execution duration. The power dissipation or consumption (P) of CMOS based processor is

$$P = C_{\text{eff}} * V^2 * f \quad (5)$$

(see [8]). C_{eff} = effective load capacitance, V = supply voltage, and f = clock frequency.

Thus the energy consumption is given by

$$E = C_{\text{eff}} * V^2 * f * t \quad (6)$$

(see [9]). The total energy consumed by k processors of CMOS system is

$$E_{\text{system}} = C_{\text{eff}} * \sum_{i=1}^{k} V_i^2 * f_i * t \quad (7)$$

(see [9]). In this model, the total energy consumption of the system is the sum of the energy consumed by the server node and client nodes.

$$E_{\text{system}} = E_{\text{server}} + E_{\text{clients}}. \quad (8)$$

The power consumption of the individual thread/task for EEAPS can be computed as

$$P_{i,k} = \left[(\beta_k - \gamma_k) * \left(\frac{M}{\alpha_i}\right)\right] + \gamma_k, \quad (9)$$

where β is busy power, γ is idle power, α_i is computational capacity for the corresponding task τ_i, i is task ID, k = processor ID, M = million instructions per second for the corresponding task = (PCS $*$ IEPC)/10^6, PCS is Processor Clock Speed, and IEPC is number of instructions executed per cycle.

Then the energy consumption of the task is given by

$$E_{\tau_i} = P_{i,k} * C_i \quad (10)$$

and C_i represents task execution time.

The sum of the power consumption of the tasks will give total power consumption of the clients. It can be computed as

$$P_{\text{clients}} = \sum_{i=1}^{N} \sum_{k=1}^{M} P_{i,k}. \quad (11)$$

Here M represents number of client nodes present in the system. The total energy consumption of the clients is thus calculated as

$$E_{\text{clients}} = \sum_{i=1}^{N} \sum_{k=1}^{M} P_{i,k} * C_i. \quad (12)$$

Table 2: Task set with attributes.

Task/thread ID	Execution time (C_i)	Time period (T_i)
Thread 1	9 msec.	20 msec.
Thread 2	11.3 msec.	25 msec.
Thread 3	21 msec.	X
Thread 4	23.5 msec.	X
Thread 5	6 μsec.	31 μsec.
Thread 6	40 μsec.	60 μsec.
Thread 7	3 msec.	9 μsec.
Thread 8	25 μsec.	45 μsec.

X represents aperiodic/sporadic task.

Then the total energy consumed by the system can be computed as

$$E_{\text{Total}} = E_{\text{server}} + \sum_{i=1}^{N} \sum_{k=1}^{M} P_{i,k} * C_i. \quad (13)$$

4. Hardware Details

It consists of four ARM7 LPC2148 microcontrollers. It follows client-server model. Processor4 itself acts as server. The server node acts as shared memory and the other three nodes act as clients. The communication between the clients is not possible in the present architecture.

The real-time application of health monitoring system consisting of eight real-time tasks is considered. Server node holds the information of all tasks and it runs task allocation algorithm in order to reduce the energy consumption of the system. The client-server communication is established using ZigBee.

The real-time tasks are as follows:

(i) Patient body temperature measuring task using an ADC-channel I

(ii) Motor control task to adjust room temperature with an ADC-channel II

(iii) EMG sensor task to sense the muscular activities of the patient with an ADC-channel III

(iv) ECG sensor task to sense the heart activities of the patient with an ADC-channel IV

(v) LED ON/OFF task to indicate the power availability

(vi) LCD display task to display patient health status

(vii) Power switch ON/OFF task to trigger the buzzer

(viii) Buzzer task to generate emergency signal

Real-time tasks with their attributes are shown in Table 2. All tasks except task3 and task4 are periodic. These two tasks may be aperiodic or sporadic in nature depending on the patient health condition.

The frame format of ZigBee is shown below.

Header (1 byte)	Data (n byte)	EOF (1 byte)

Here header represents address/ID of the client. EOF indicate end of frame. ZigBee provides acknowledged transmission of data.

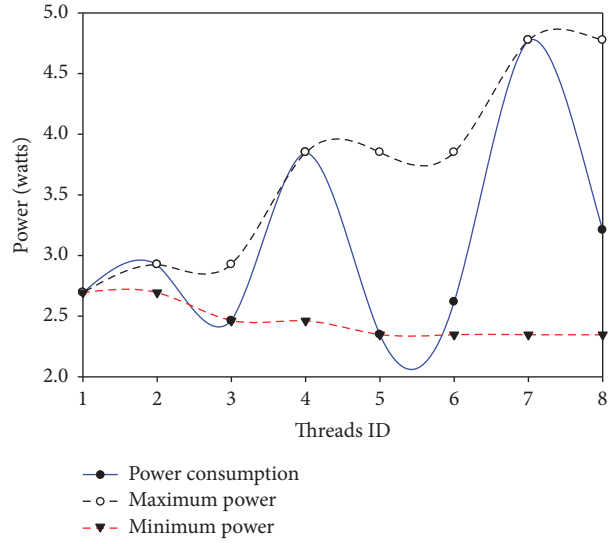

Figure 1: Power analysis.

5. Real-Time Simulator

Real scheduling issues are far removed from theoretical assumptions; simulations analyse and address them. To enable the investigation of scheduling algorithm, we used an enthusiastic tool: RTSIM, a real-time scheduling simulator. It is available in open source platform and user friendly module. The main advantages of this compiler are the computational effective nature and optimization of the workload through its modules MetaSim, rtlib, ctrlib, and jtracer. Each module has its own contribution; all these things will help the user to understand, simulate, and analyze the scheduling algorithms in a better way.

RTSIM is built on the MetaSim framework [19], which grants general interface for the prioritized event queue. In this simulation it is divided into ticks. It offers three key notions: processors accomplished by a *kernel* are owed to *tasks* by a *scheduler*. Here, we used modified kernel notion for multiple processors and task notion for modelling the tasks. We applied roots of the scheduler notion for our scheduling algorithms. We made comprehensive simulation of our algorithm and the outcomes are offered in the subsequent section.

6. Results

The proposed algorithm is simulated by using the RTSIM simulator.

Table 3 represents power consumption of the threads by considering processor parameters like busy power, idle power, computational capacity, number of processors, and the number of cycles per instruction into account. The graphical representation of the thread's power consumption and their minimum and maximum power requirements are shown in Figure 1. Here the y-axis represents the power in terms of watts and x-axis represents the threads ID.

Table 3: Power consumption of the threads.

Thread ID	Number of cycles for instruction	Power consumption in watts	Max power	Min power
Thread 1	8	2.694	2.694	2.694
Thread 2	6	2.925	2.925	2.694
Thread 3	12	2.462	2.925	2.462
Thread 4	16	3.850	3.850	2.462
Thread 5	3	2.347	3.850	2.347
Thread 6	9	2.617	3.850	2.347
Thread 7	2	4.775	4.775	2.347
Thread 8	5	3.210	4.775	2.347

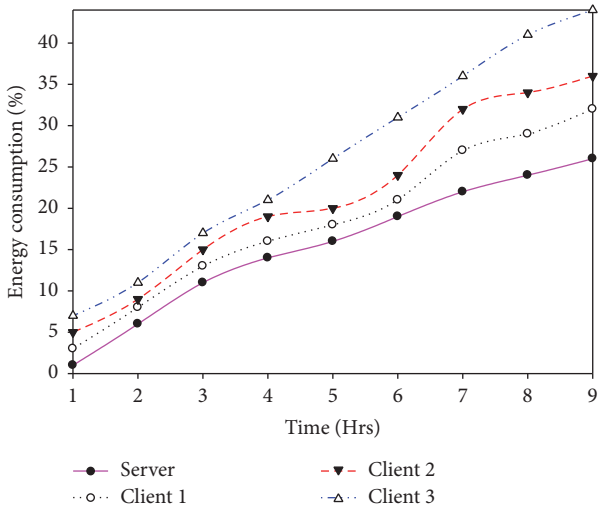

Figure 2: Energy consumption of the processors.

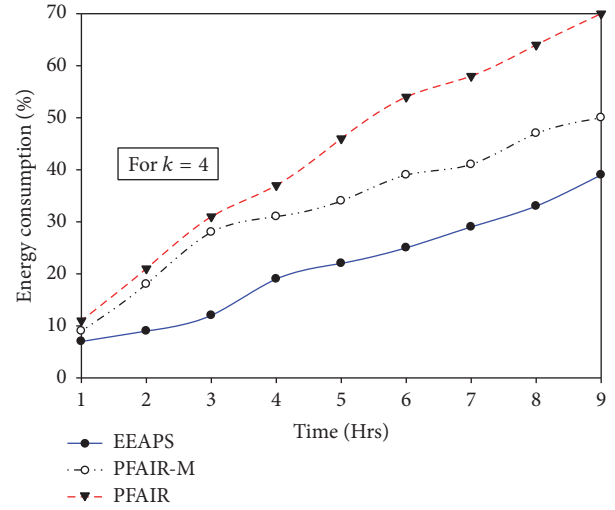

Figure 3: Comparison of EEAPS energy consumption with Pfair algorithms.

The task set is simulated for the EEAPS algorithm and the energy consumption of the processor nodes is displayed in Figure 2.

The energy consumption results of the EEAPS scheduler are compared with the existing optimal Pfair and modified Pfair schedulers. This is displayed in Figure 3. The graph clearly shows that EEAPS reduces 10% of energy consumption with respect to Pfair-M and 30% of energy consumption with respect to Pfair algorithm.

7. Conclusion

The proposed energy efficient scheduling algorithm EEAPS for multiprocessor systems calculates the energy requirement of each task. Based on this the tasks are allocated to the processors through the auctioning mechanism. This model reduces the energy consumption of the system. It is more effective compared to the existing scheduling algorithms. It is so flexible and compatible to different task set for different processors. It is optimal for periodic and aperiodic tasks but it is not for sporadic tasks. This model allocates the dedicated processor to handle the sporadic tasks without missing the deadline.

Competing Interests

The authors declare that they have no competing interests.

References

[1] G. Anastasi, M. Conti, M. Di Francesco, and A. Passarella, "Energy conservation in wireless sensor networks: a survey," *Ad Hoc Networks*, vol. 7, no. 3, pp. 537–568, 2009.

[2] R. Rajagopalan and P. K. Varshney, "Data-aggregation techniques in sensor networks: a survey," *IEEE Communications Surveys and Tutorials*, vol. 8, no. 4, pp. 48–63, 2006.

[3] K. Akkaya and M. Younis, "A survey on routing protocols for wireless sensor networks," *Ad Hoc Networks*, vol. 3, no. 3, pp. 325–349, 2005.

[4] M. Azeem, M. I. Khan, S. U. Khan, and W. Gansterer, "Efficient scheduling of sporadic tasks for real-time wireless sensor networks," *IET Wireless Sensor Systems*, vol. 5, no. 1, pp. 1–10, 2015.

[5] A. S. Pillai and T. B. Isha, "EC-A: a task allocation algorithm for energy minimization in multiprocessor systems," *Middle East Journal of Scientific Research*, vol. 18, no. 6, pp. 779–787, 2013.

[6] S. Baskaran and P. Thambidurai, "Energy efficient real-time scheduling in distributed systems," *IJCSI International Journal of Computer Science Issues*, vol. 7, no. 3, pp. 35–42, 2010.

[7] E. Ilavarasan and R. Manoharan, "High performance and energy efficient task scheduling algorithm for heterogeneous mobile computing system," *International journal of computer science & information Technology*, vol. 2, no. 2, pp. 10–27, 2010.

[8] K. Funaoka, S. Kato, and N. Yamasaki, "Energy-efficient optimal real-time scheduling on multiprocessors," in *Proceedings of the 11th IEEE Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing (ISORC '08)*, pp. 23–30, Orlando, Fla, USA, May 2008.

[9] J. Zhuo and C. Chakrabarti, "Energy-efficient dynamic task scheduling algorithms for DVS systems," *ACM Transactions on Embedded Computing Systems*, vol. 7, no. 2, article 17, 2008.

[10] J.-M. Chen, K. Wang, and M.-H. Lin, "Energy efficient scheduling for real-time systems with mixed workload," in *International Conference on Embedded and Ubiquitous Computing: EUC 2007: Embedded and Ubiquitous Computing*, Springer, Berlin, Germany, 2007.

[11] S. K. Baruah, N. K. Cohen, C. G. Plaxton, and D. A. Varvel, "Proportionate progress: a notion of fairness in resource allocation," *Algorithmica*, vol. 15, no. 6, pp. 600–625, 1996.

[12] J. Goossens, S. Funk, and S. Baruah, "Priority-driven scheduling of periodic task systems on multiprocessors," *Real-Time Systems*, vol. 25, no. 2-3, pp. 187–205, 2003.

[13] S. K. Baruah, J. E. Gehrke, and G. C. Plaxton, "Fast scheduling of periodic tasks on multiple resources," in *Proceedings of the IEEE 9th International Parallel Processing Symposium*, pp. 280–288, Santa Barbara, Calif, USA, April 1995.

[14] A. Srinivasan, P. Holman, J. H. Anderson, and S. Baruah, "The case for fair multiprocessor scheduling," in *Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS '03)*, Nice, France, April 2003.

[15] I. Stoica, H. Zhang, and T. S. E. Ng, "A hierarchical fair service curve algorithm for link-sharing, real-time, and priority services," *IEEE/ACM Transactions on Networking*, vol. 8, no. 2, pp. 185–199, 2000.

[16] A. Chandra, M. Adler, and P. Shenoy, "Deadline fair scheduling: bridging the theory and practice of proportionate-fair scheduling in multiprocessor servers," in *Proceedings of the 21st IEEE Real-Time Technology and Applications Symposium*, 2001.

[17] J. H. Anderson and A. Srinivasan, "Mixed Pfair/ERfair scheduling of asynchronous periodic tasks," *Journal of Computer and System Sciences*, vol. 68, no. 1, pp. 157–204, 2004.

[18] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *Journal of the ACM (JACM)*, vol. 20, no. 1, pp. 46–61, 1973.

[19] Retis lab, scuola superiore sant'anna, The RTSIM project, http://rtsim.sssup.it.