A Lazy DVS Approach for Dynamic Real Time System

Smriti Agrawal
Department of Information Technology,
Chaitanya Bharathi Institute of Technology, Hyderabad, India.
agrawal.smritil@gmail.com

Abstract

This paperpresents a LazyDVS scheduling algorithm that offers higher acceptance ratio (number of tasks able to meet their deadline is to number of tasks appearing) with equivalent orlesser energy consumption for battery powered dynamic real time system. It is modeled for aperiodic workload by utilizing the concept of late start and dynamic voltage scaling. In this paper,with a motivational example we show that existing approaches have poor energy and aperiodic workload management, thus, are not suitable for scheduling aperiodic workload on limited source of energy. Based on this motivation we propose a Lazy DVS scheduling algorithm that tradeoffs between timing constraint and the available energy to provide better acceptance ratio with lower energy. The extensive examples and simulation results illustrate that our approach can effectively improve the acceptance ratio while consuming lower energy.

*Keywords:*Real time systems, energy aware scheduling, aperiodic, dynamic voltage scaling, quality of service (QoS).

1. Introduction

Reduction in energy consumptionis one of the main factors for designing battery powered real time embedded systems. These systems need portability so must be compact and lightweight. The advancement in technology has reduced the chip area but the energy requirement has increased correspondingly.Such systems if deploy bigger battery it resultsin increase in their size as well as cost which in turn severely limits the system's lifespan and portability. Thus, designing an energy aware real time embedded systems is a possible answer.

Significantwork on energy aware scheduling based on Dynamic Voltage Scaling has been done. However, most of the algorithms [1, 3, 5, 6, 9, 15], are targeted for periodic tasks which assumesthat all temporal requirement such as release time, execution time and deadline are known in advance. Such systems are traditionally scheduled in offline where they provide a guaranteed quality of service both in terms of number of tasks meeting their deadlines as well as duration of energy backup. On the other hand, many real time applications involve aperiodic tasks. An aperiodic task isevent based soits temporal requirement is not known in advance, thus,itsscheduling cannot be done offline. However, an aperiodic task once accepted must meet finish by its deadline. An improved quality of service by accepting more number of aperiodic taskssuch that they finish within their deadline with the available amount of energy is the key parameter in scheduling such systems. This paper, targets aperiodic task based systems andstrives to improve the quality of service with limited energy available by using Dynamic Voltage Scaling (DVS).

Existing strategies[2, 4, 7, 8]that strive for reduction of energy consumption of aperiodic tasks are either time or energy conformist. The authors [2, 4] suggested a greedy based approach. In this technique whenever an aperiodic task arrives if the time and energy permits it is accepted and scheduled as soon as possible. However, if the system is incapable of completing it within its deadline then it is rejected. This approach has a high rejection rate because all the energy is consumed by the tasks arriving early, leaving little or no energy for its successors. The authors [7] refined it and suggested a lazy approach in which the system does not greedy schedule the ready task and thus, consume all the available energy rather it will accumulate the ready task up to maximum slack time and then select a task to be accepted or rejected. The shortcoming of this approach is that although it saves energy but tasks are rejected due to time constraints. A DVS based scheme is suggested by authors[8, 10, 11, 12, 13, 14], such that whenever the time permits the lowest energy consuming speed is assigned. The motivational example in the following section illustrates these approaches.

In this paper, we aim to improve the quality of service (QoS) by accepting more number of aperiodic tasks as well as minimize the energy consumption leading to elongated operating time of the battery.

The rest of the paper is organized as follows: in section II, we describe our preliminary, followed by motivational example and system model. Sections III discuss our contribution. Section IV elaborates our proposed approach followed by results and analysis in section V. Finally, paper concludes with section VI.

2. Motivational Example

In this section, we present a motivational example which will illustrate the existing techniques.

Example: Consider a real time system working on a DVS processor with speed levels $\{s/2, 3s/4, s\}$. The energy available is 1.7 units and is replenished at every window of size 7. The system consumes 1 unit of energy per unit time at maximum speed. During a snap shot suppose three aperiodic tasks, $(\tau_i = \{(a_i, e_i(s), d_i): (0, 1.5, 8), (1, 1, 3), (2.5, 0.5, 3.2)\})$ where a_i is the arrival time of task τ_i while $e_i(s)$ is the computation time required at maximum speed s and d_i is the absolute deadline) arrive which need to be scheduled.

The schedule as computed by the existing techniques is described in the following subsections:

Greedy approach (GA):

At time t=0 when the task τ_1 arrives, the scheduler calculates the energy requirement as 1.5 and finish time as 1.5. The available energy in the system at this point of time is 1.7, i.e., greater than that required by this task and no other higher priority task is available hence, this task is accepted and starts execution.

At time t=1, when task τ_2 arrives, the scheduler estimates that it energy requirement is 1 while available energy after the task τ_1 is only 0.2. Hence, it rejects task τ_2 due to unavailability of sufficient energy. Thus, task τ_1 finishes at time t=1.5.

At time t=1.5 no task is in the ready queue so the system switches to idle state and remains there until time t=2.5, when task τ_3 arrives. At this point the scheduler again estimates the energy requirement as 0.5 while the available energy is merely 0.2, hence, it rejects τ_3.

Thus, the greedy approach is able to schedule only one task out of three aperiodic tasks (33%) that arrived. This schedule is plotted in Figure 1(a). The following subsection describes the Lazy approach adopted to improve the performance of this scheme.

Lazy Approach (LA):

Lazy approach as described in Section I delays the scheduling up to the maximum slack time, preserves energy for likely arrival for few more, and then decides how many it can accept. In the above example when task τ_1 arrives at time t=0 it does not accepts/rejects it. Rather it estimates the slack time and accordingly calculates the start time as, 8-1.5=6.5. Thus, the system remains idle until time t=1 when task τ_2 arrives.

At time t=1, again it estimates the best start time subject to completion of this task as well as not missing the deadline of task τ_1, as 3-1=2. Thus, the system remains idle till time t=2

At time t=2, since, no other task has arrived, hence, it will accept task τ_2 and start its computation. At time t=2.5 a lower priority (based on Earliest Deadline First) task arrives. Thus, task τ_2 continues its execution and finishes at its deadline of 3. It consumesa total of 1 unit of energy and the system is left with 0.7 units.

At time t=3, the scheduler selects the task τ_3, since it has the highest priority. It estimates its energy requirement as 0.5 while 0.7 units is available, hence, task τ_3 will be feasible from the energy point of view. However, the time available up

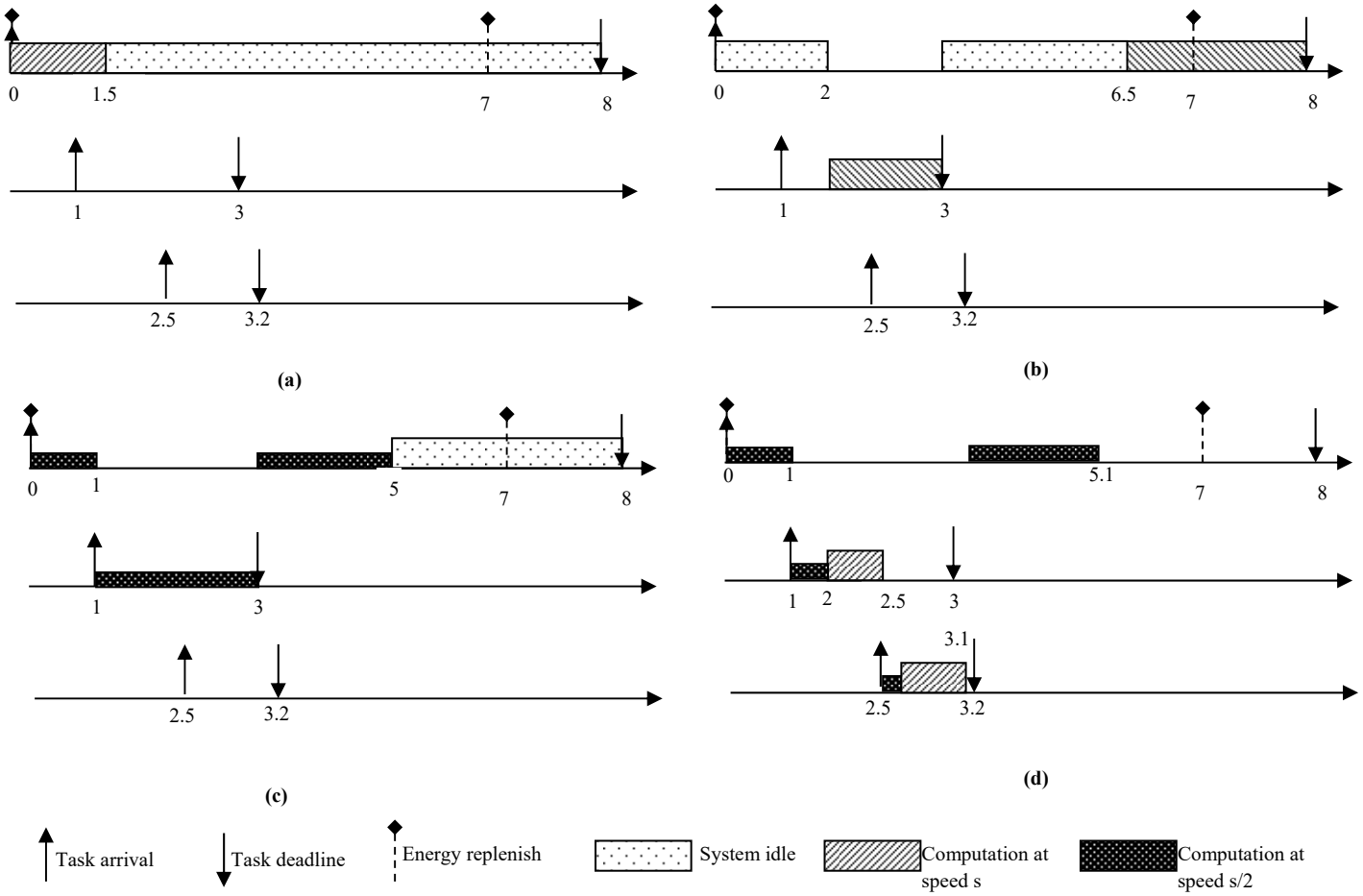


Figure 1: Schedule for example ($\tau_i = \{(a_i, e_i(s), d_i)$: $(0, 1.5, 8), (1, 1, 3), (2.5, 0.5, 3.2)\}$) according to (a) greedy approach (b) Lazy approach (c) DVS based approach and **(d) ProposedLazy DVS**

to its deadline is only 0.2 while it requirement at highest speed is 0.5. Hence, task τ_3 will fail to meet its deadline. Thus, the scheduler reject it and system remains idle until time t=6.5.

At time t=6.5, since, both time and energy permits, scheduler will accept task τ_1.

Thus, Lazy approach is able to accept two tasks (66%), as compared to greedy approach that could schedule only one (33%). The schedule for this interval is plottedin the figure 1(b). The following subsection describes the DVS based approach for the same example.

DVS based Approach (DVS)[8]:

The authors [8] suggested a DVS approach, which advocates reduction in the computation speed to the lowest feasible for Sporadic Tasks. It lowers the energy consumption of an executing task, on the job by job basis and adjusts the processor frequency when the jobs are released. Thesame technique can be applied to aperiodic tasks. Thus, for the above example as illustrated in the figure 1(c), when task τ_1 arrives lowest speed at which it would be feasible is $s/2$. Thus, task τ_1 is accepted and computation begins at speed of $s/2$.

At time t=1, when task τ_2 arrives, task τ_1 has completed 33% of its total computation at speed $s/2$. However, slower computation speed employed by task τ_1, saves energy, hence, task τ_2 is accepted and starts executing.

At time t=2.5, task τ_3 arrives. Task τ_2, has performed 75% of its computation up to this point. It further requires 0.5 unit of time at speed s/2, while at speed srequirement is of 0.25. Thus, task τ_2 can finish earliest by time t=2.75. The computation time requirement of task τ_3 at maximum speed is 0.5, hence, it can finish time earliest by 3.25, missing its deadline of 3.2.The scheduler will thus, reject task τ_3 and continue with task τ_2 at lowest speed.

At time t=3, when task τ_2 completes only ready task is task τ_1, which is scheduled and completes at time t=5, and system is idle. The DVS based approach is able to reduce the energy consumption, but still has to reject tasks due to unavailability of time (33% rejection rate).

The example clearly demonstrates that the above approaches are either time- or energy-constrained. This is because they either waste time doing nothing or they consume too much of energy leaving insufficient resources for the upcoming tasks. This paper strives to balance between the two and suggests an approach that will produce higher quality of service.

3. System Model

This system deals with energy minimization of random arrival pattern aperiodic tasks and is able to operate at different speed level.

The following considerations are made:

1. System consists of n independent aperiodic tasks τ_1, τ_2, $\tau_3 \dots \tau_n$. Each task τ_i has the attributes, (a_i) is the arrival time, $e_i(s_{\mathcal{N}})$ is the worst-case execution time at maximum speed level $(s_{\mathcal{N}})$ and deadline(d_i), which are known only after its release. A task once accepted is guaranteed to complete within its deadline and not violating the deadlines of any previously accepted tasks.
2. System is uni-processor with a set of independent, preemptive tasks.
3. Dynamic priority scheduling algorithm Earliest deadline first (EDF) is used.
4. DVS processor can operate at $\mathcal{N}+1$ discrete voltage levels, i.e., $V = \{v_{slp}, v_1, v_2, v_3 \dots v_{\mathcal{N}}\}$ where each voltage level is associated with a corresponding speed from the set $S = \{s_{slp}, s_1, s_2, s_3 \dots s_{\mathcal{N}}\}$ .The speed s_1 is the lowest operating speed level whereas maximum speed is represented as $s_{\mathcal{N}}$. A processor can be in either activeor sleep state. In the active state the processor can run at any of the speed levels between $s_1$to $s_{\mathcal{N}}$, while in the sleep state it will function at speed s_{slp}.The power consumption is proportional to the cube of the operating speed.

The figure 2 depicts the energy aware scheduling scenario. Theenergy storage is the place to store energy and its capacity is denoted as C. Instead of making the entire stored energy available to the processor at one time, the window based energy aware system will ensure that in no window the total consumed energy is greater than E_{rep}. In other words, at every p_{rep} the available energy for the execution isE_{rep}, thus, all the tasks in the window of $(0, p_{rep})$ must be scheduled consuming at most E_{rep} of energy. At any time the energy available is represented as $E_{av}(t)$. At time $t = 0, p_{rep}, 2p_{rep}, 3p_{rep} \dots np_{rep}$ the energy available$E_{av}(t) = E_{rep}$. The symbols used are summarized in the table 1.

Table 1: Symbol Table	
$E_{av}(t)$	Energy available at time t
E_{rep}	Amount of energy replenished
a_j	Arrival time of jth taskτ_j
d_j	Deadline of jth taskτ_j
W_i	ith number of energy window
W_i^S	Starting point of ithenergy window
W_i^E	Ending point of ithenergy window
ε_{pslp}	Energy consumed per unit time by the processor in the sleep state
$\varepsilon(s_i)$	Energy consumed per unit time by the processor when running at a speed s_i ($\varepsilon(s_i) = Ks_i^3$ where K is a constant)
E_{reqj}	Energy required by task $\tau_j$for computation
C	Total Energy storage capacity
$R_i(s_j)$	Response time of task $\tau_i$at speed (s_j)
St_i	Start time of the ith task τ_i
s_{ai}	Speed assigned to task τ_i

The following section presents the proposed technique to reduce the energy consumption for the system modelled in this section.

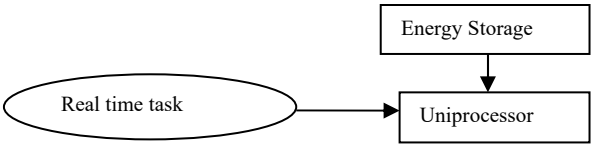


Figure 2: Energy aware real time scheduling Scenario

4.Proposed Lazy DVS

The scheduling of the aperiodic tasks is best effort service as the system requirements are dynamic. Such systems can be scheduled online only. However, the scheduling will combine both dynamic priority scheduling technique for priority assignment and Lazy DVS technique for the start time estimation of a task. This scheduling framework can be seen in Figure 3.

The motivational example in Section 2, clearly illustrates the various techniques available in literature for scheduling of the aperiodic task with energy considerations. On closer look, it can be observed that the existing techniques fail due to one or more of the following reasons:

1. They overstress the system and consume all energy available (greedy approach), leaving little or no energy for the forth coming tasks.
2. They postpone the scheduling too far, so the upcoming tasks have little or no time available (Lazy approach).
3. They execute the available tasks at such low speed that they leave little or no room for the random tasks that arrive (DVS based approach).

This paper tries to balance these two orthogonal conditions by the following rules:

R1: Postpone the execution of a task up to a *start time* ($St_i = d_i - R_i(s_{ai})$, as explained in the following paragraph), while keeping in account the feasibility of already accepted tasks.

R2: Unlike the Lazy approach, do not leave the system idle instead schedule the ready task with minimum energy consumption.

R3: If a task is not feasible at maximum speed due to its energy requirement at that speed, select a lower (than maximum) energy consuming speed, if its deadline permits.

As stated above, whenever a task $\tau_i$arrives, a start time for it and its lower priority tasks, already accepted, is estimated as follows:

$\left(St_i = d_i - R_i(s_{ai})\right)$, where St_i, d_i are the start time and deadline of the task $\tau_i$and $R_i(s_{ai})$ is the response time of the task at the assigned speed, computed as $R_i(s_{ai}) = e_i(s_{ai}) + \sum_{h\in H} e_h(s_{ah})$ where H is the set of higher priority tasks already accepted. In case the estimates start time is negative indicating that $d_i > R_i(s_{\mathcal{N}})$, this task has to be rejected, since it cannot be completed within its deadline. Further, the energy required is also estimated as $E_{reqi} = e_i(s_{ai})\varepsilon(s_{ai})$. For a task to be feasible, in terms of energy, its required energy E_{reqi}at all times should always be less than or equal to available stored energy in $[St_i, d_i]$ interval, i.e., $\varepsilon(s_{ai}) \leq E_{av}(t) \forall\ t\epsilon[St_i, d_i]$.Thus, a task is accepted if and only if the deadline and energy requirement of this task along with the lower priority tasks previously accepted can be met. The algorithm for the proposed Lazy DVS is stated as follows:

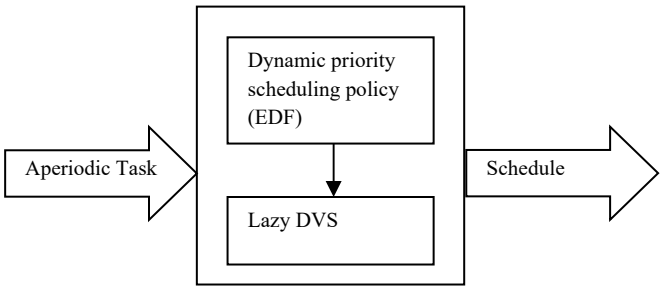


Figure 3: Energy aware schedulingframework

// Input: Tasks are inserted in to the priority ready queue based on their start time.

Algorithm LazyDVS()

1. ***While(ready queue is empty)***
   a. ***sleep and do nothing***
2. ***While (ready queue not empty)***
   Do
   a. ***Select a task τ_i with the earliest start time from the ready queue.***
   b. ***If (start time of τ_i= current time)***
      i. ***Execute the task τ_i at speed assigned s_{ai} till***
         A. ***It finishes, then go to step 2.***
         B. ***A new taskarrives, then save the status and go to step 3.***
         C. ***A higher priority task τ_h (based on EDF) preempts it, then save its status and go to step 2.b.iwith $\tau_i = \tau_h$.***
   Else
      i. ***Select the task τ_j with the highest priority based on EDF***
      ii. ***Execute the task τ_j at lowest speed s_1 till***
         A. ***It finishes, then go to step 2.***
         B. ***A new task arrives, then save the status and go to step 3.***
         C. ***A higher priority task τ_h (start time of τ_h = current time) preempts it, then save its status and go to step 2.b.i with $\tau_i = \tau_h$.***
   Repeat
   Go to step 1
3. ***For every task $\tau_i$arriving with attributes $(a_i, e_i(s_{\mathcal{N}}), d_i)$***
   Do
   a. ***Assign the speed $s_{ai} = s_{\mathcal{N}}$***
   b. ***Estimate the start time as $St_i = d_i - R_i(s_{ai})$***
   c. ***If (St_i<0)***
      i. ***Reject it and goto step 1***
   d. ***Else***
      i. ***Estimate the energy requirement $E_{reqi} = e_i(s_{ai})\varepsilon(s_{ai})$***
      ii. ***If($\varepsilon(s_{ai}) \leq E_{av}(t) \forall\ t\epsilon[St_i, d_i]$)***
         A. ***Insert the task the ready queue and goto step 2.***
         Else
      A. ***If ($s_{ai} = s_1$)***
         a. ***Reject it and go to step 2***
         Else
         a. ***Assign next lower speed level and go to step 3.b.***
   Repeat
   Go to step 1

Thus, scheduling the motivational example (Section 2) using the proposed LazyDVS algorithm. Attime t=0 when first aperiodic task $\tau_1$arrives, its start time is estimated to be 6.5 (as per the rule R1, step 3 of the algorithm), which means that the system should remain idle till time 6.5 if no other task arrives in the meanwhile and finally start execution of the task τ_1 at 6.5 at maximum speed. However, the proposed lazy approach as per rule R2(Step 2 b of the algorithm) will start execution of this task at the lowest possible speed so as to consume least energy as well as utilize the otherwise wasted time.

At time t=1 when task τ_2 arrives, the system has already completed 33% of the task τ_1 computational requirement. The start time for τ_2 is now estimated to be 2 by rule R1. Thus, the system should remain idle until time t=2, however, as per rule R2, task τ_2 will be executed with the lowest possible energy consumption. However, at time t=2, as its start time has occurredτ_2 will speed up to maximum speed and complete by time t= 2.5.

At time t=2.5 task τ_3 will arrive and its start time as per rule R1 is estimated to be 2.7, hence, it will execute consuming lowest energy from time 2.5-2.7. However, at time t=2.7, its start time, it will speedup to complete at 3.1. From time t=3.1 task τ_1 will start executing consuming lowest energy and will finish at time t=5.1, still saving 0.275 units of energy. Thus, the proposed approach is capable of accepting all the three tasks (100%) arriving in this example as compared to the existing approachesthat are able to accept at most 66% of the incoming load. Further, it also saves energy, 0.275 units in the current example.

The next section deals with performance measurement ofthe proposed LazyDVS approach through simulations.

5. Simulation Results and Discussion

In this section, simulations on synthesized tasks are performed to evaluate the performance of the proposed Lazy DVS scheduling. The processor is capable of voltage and frequency scaling. The key parameters for performance measurment are average energy consumption and acceptence ratio(no of tasks accepted upon no of incoming tasks). Aperiodic tasks were generated by the exponential distribution using with inter arrival time (1/λ) and service time (1/μ) with parameters λ and μ.Simulation is run for 10000 aperiodic tasks.We implemented the following approach for performance evaluation in this case:

Greedy Approach(GA): which schedule the jobs atmaximum avilable speed on greedy basis.

***Lazy Approach (LA)*:** which schedule the jobs at maximum avilable speed and defered their startitng time on the basis of time as well as avilable energy.

Dynamic Voltage Scaling for Sporadic Tasks (DVSST [8]):starts with a minimum possible frequency-scaling factor, and scales the processor frequency up and down depending when jobs are released.

Proposed Lazy Dynamic Volatge Scheduling (LazyDVS): an online speed assignment algorithn for aperiodic tasks that assignes maximum speed intially but utilizes any available time toexecute with least energy consumption for all ready jobs at current time t.

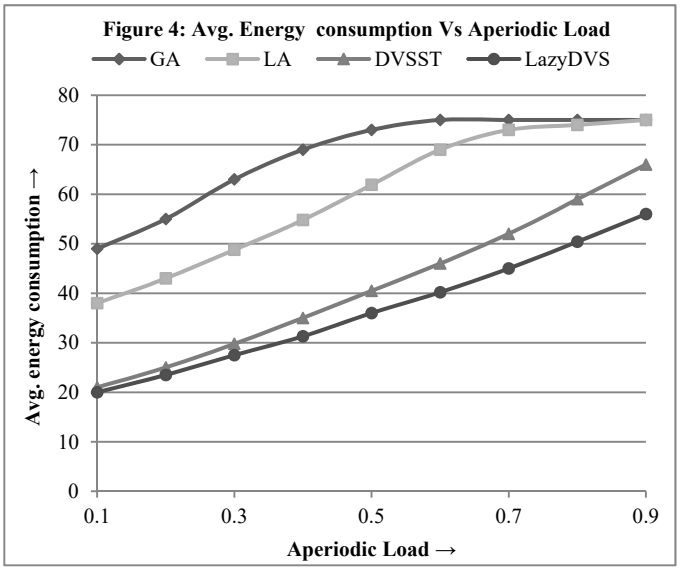
In the following section we measure the effect of variation in aperiodic load on the average energy consumption and acceptense ratio of aperiodic task.

Effect of load on Average energy consumptionand acceptance ratio of aperiodic task:

The effect of load on the average energy consumption and acceptence ratio can be seen from the figure 4and figure 5 respectively.

Figure 4, compares the average energy consumption of *Greedy Approach (GA), Lazy Approach (LA), Dynamic Voltage Scaling for Sporadic Tasks (DVSST) and Lazy Dynamic Volatge Scheduling (LazyDVS)* when aperiodic load varies from 10% to 90%. It is observered thatas load increases, average energy consumption also increasesfor all the approaches. However, when it is high say 70% to 90% the proposed LazyDVS approach providesalmost 13% reduction in average energy consumption over existing DVSST approach. This is because LazyDVS is better utilized when occurrence of more aperiodic task is higher. On the other hand, when the aperiodic load is varied from 10% to 40% proposed approach has approximately 6% reduction in average energy consumption over DVSST. This is due to the most of the time aperiodic will execute at same speed level in both approach.

Figure 5,compares the performance of *Greedy Approach (GA), Lazy Approach (LA), Dynamic Voltage Scaling for Sporadic Tasks (DVSST) and Lazy Dynamic Volatge Scheduling (LazyDVS)*when aperiodic load varies from 10% to 90% in terms of acceptance ratio. It is observedthatas load increases acceptence ratio decreases for all approaches. However, when the aperiodic load is high say 70% to 90% the proposed Lazy DVS accepts approximately28% more tasks than simple Lazy approach (LA) without DVS, approximately 35% more as compared to Greedy approach (GA) and as compared to DVSST itis 10%. This is because limited energy



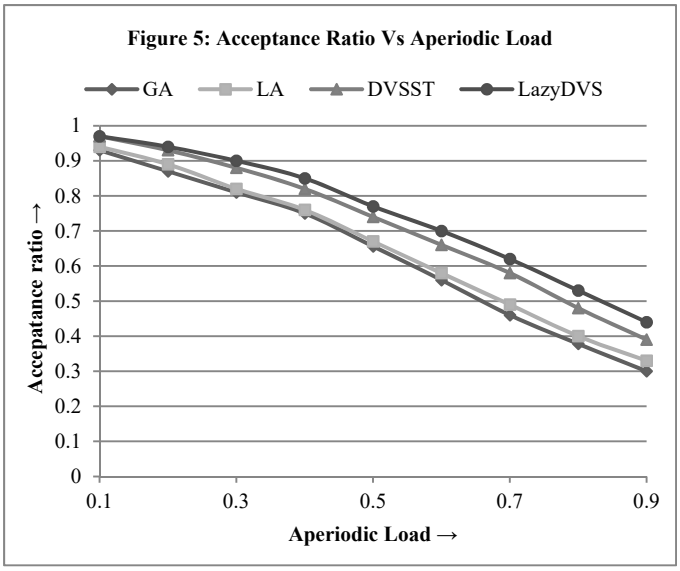


Figure 5: Acceptance Ratio Vs Aperiodic Load

is avilable, however, without DVS based approaches (GA and LA) always execute the task at maximum avilable speed so stored energy is consumed earliar and more chance to reject task due to energy constraints. However, DVSST switches the task to lowest possible speed to save energy may waste so much of time doing one job that there is little or no room left for the upcoming tasks, leading to poor acceptance ratio. While at lower aperiodic load (10% to 20%) without all approach accept most of the aperiodic tasks as sufficient amount of energy and time is avilable.

6. Conclusion

In this paper, we presented a Lazy DVS scheduling algorithm that focused on obtaining maximal utility while respecting the limited amount of available energy for aperiodic tasks.A new aperiodic task is accepted only and only if it can be guaranteed to finish within its deadline with the available energy without violating the deadlines of the previously accepted tasks. For meeting the timing as well as energy requirement, we proposed a set of rules which when followed save energy. The proposed algorithm improvesthe performance both in terms of energy saving and acceptance ratio.

The examples and simulation studies has carried out. It has been observed that the proposed scheduling algorithm reduce the overall average energy consumptionof aperiodic tasks is approximately 13 % at aperiodic load varied from 70% to 90% while 6% at lower aperiodic load varied from 10% to 50%. When the aperiodic load is high say 70% to 90% our proposed approach LazyDVS could accept approximately 10% more task than simple Dynamic Voltage Scaling Algorithm and accept 35% more task as compare to Greedy approach without DVS (GA). Thus, extensive simulation and illustrative example shows that our proposed approach is capable of performing better in terms of average energy consumption of aperiodic task as well as acceptance ratio of aperiodic task.

7. References

[1.] H. Aydin, R.G. Melhem, D. Mosse´, and P. Mejı´a-Alvarez, “Power-Aware Scheduling for Periodic Real-Time Tasks”, IEEE Trans.Computers, vol. 53, no. 5, pp. 584-600, May 2004.

[2.] I. Hong, M. Potkonjak, and M.B. Srivastava, “On-Line Schedulingof Hard Real-Time Tasks on Variable Voltage Voltage Processor,”Proc. Int’l Conf. Computer-Aided Design, pp. 653-656, 1998.

[3.] P. Mejı´a-Alvarez, E. Levner, and D. Mosse´, “Adaptive SchedulingServer for Power-Aware Real-Time Tasks,” ACM Trans. EmbeddedComputing Systems, vol. 3, no. 2, pp. 284-306, 2004.

[4.] V. Sharma, A. Thomas, T. Abdelzaher, K. Skadron, and Z. Lu,“Power-Aware QoS Management in Web Servers,” Proc. IEEEReal-Time Systems Symp., pp. 63-72, 2003.

[5.] P. Pillai and K.G. Shin, “Real-Time Dynamic Voltage Scaling forLow-Power Embedded Operating Systems,” in Proc. 18th Symp.Operating Systems Principles, pp. 89-102, 2001.

[6.] Y. Zhu and F. Mueller, “Feedback EDF Scheduling ExploitingDynamic Voltage Scaling,” Proc. IEEE Real-Time and EmbeddedTechnology and Applications Symp., pp. 203-212, 2004.

[7.] A. Sinha and A.P. Chandrakasan, “Energy Efficient Real-TimeScheduling,” Proc. Int’l Conf. Computer-Aided Design, pp. 458-470,2001.

[8.] A. Qadi, S. Goddard, and S. Farritor, “A Dynamic Voltage ScalingAlgorithm for Sporadic Tasks,” Proc. IEEE Real-Time SystemsSymp., pp. 52-62, 2003.

[9.] Y. Shin and K. Choi, “Power Conscious Fixed Priority Schedulingfor Hard Real-Time Systems,” Proc. Design Automation Conf.pp. 134-139, 1999.

[10.] Y. Doh, D. Kim, Y.-H. Lee, and C. M. Krishna. ConstrainedEnergy Allocation for Mixed Hard and Soft Real-Time Tasks. InProc. of Int. Conf. on Real-Time and Embedded Computing Systemsand Applications, pages 533–550, 2003.

[11.] W. Yuan and K. Nahrstedt. Integration of Dynamic VoltageScaling and Soft Real-Time Scheduling for Open Mobile systems. InProc. of Int. Workshop on Network and Operating Systems Supportfor Digital Audioand Video, pages 105–114, 2002.

[12.] D.Shin and J.kim “Dynamic voltage scaling of periodic andaperiodic tasks in priority-driven systems” In Proc. of Asia and SouthPacific Design, pp 653-658,2004.

[13.] Y. Shin and K. Choi. Power Conscious Fixed PriorityScheduling for Hard Real-Time Systems. Proceedings of theDesign Automation Conference, pp. 134–139, June 1999.

[14.] Y.-H. Lee and C. M. Krishna,“Voltage-Clock Scaling for LowEnergy Consumption in Real-Time Embedded Systems”,Proceedingsof the Sixth Int’l Conf. on Real Time Computing Systemsand Applications, pp. 272-279, 1999.

[15.] F. Gruian, “Hard Real-Time Scheduling for Low-Energy UsingStochastic Data and DVS Processors,” Proc. Int’l Symp. LowPowerElectronics and Design, pp. 46-51, 2001.