

Real-Time Scheduling with DVS and Harvesting Energy Constraints

Mona Kumari

Assistant Professor

Department of Information Technology
Anand Engineering College, Agra

Ajitesh Kumar

Assistant Professor

Department of Information Technology
Hindustan Institute of Technology
and Management, Agra

ABSTRACT

In real-time embedded system, that must carry their own power source and cannot depends on the power outlet on the wall, apart from feasibly schedule the set of tasks, power management is also the major issue because without power the system is useless.

In this paper, we propose a harvesting aware real-time scheduling algorithm with variable speed assignment scheme to set of periodic tasks aims to reduce the energy consumption while feasibly schedule the set of periodic tasks within their deadline. This can be done by DVS(Dynamic Voltage and frequency Selection), executing the task with the speed such that a task can consume as much energy which is quite sufficient to complete it successfully within its deadline.

The example and simulation results shows that the propose approach is capable of performing better in terms of average stored remaining energy of the system as well as acceptance ratio of periodic tasks at lower periodic load.

Key Terms: Real-Time Scheduling, DVS, Energy Harvesting, Periodic Tasks, Embedded System, Power Management.

1. INTRODUCTION

In real-time system to function correctly, the system must produce a correct result within a specified time called deadline for example the anti-lock breaks on a car are a simple example of a real time computing system, the real-time constraint in this system is the time in which the breaks must be released to prevent the wheel from locking. Whatever its category (hard, soft and firm), a real-time embedded system is said to achieve energy neutral operation if its execution requirements can be supported forever despite energy limitations [8-10].

Now a day's most of the real-time embedded system executing on the platform that are mobile and carry their own power source in the form of battery and do not depend on power outlet on wall. Most of the time this mobile device remains beyond the scope of recharging their battery due to mobility or unavailability of recharging point, for example mobile video phone applications require light weight device movable across the globe [11, 13].

Most of time these light weighted devices remains beyond the scope of recharges the battery. This may be due to unavailability of recharging point or time required for recharging due to mobility. Thus, for smooth functioning of this light weight mobile device one has to facilitate it either with large capacity battery or powerful power management technique to enlarge the battery back-up time. However in some applications, replacing/recharging a battery is either costly or impractical, wireless sensor network is one of such application, where the sensor nodes are deployed in a wide wild area for environment

surveillance. Hence, ideally such a system should be designed to operate perpetually with the battery being the only energy source [1, 3].

With the advancement in battery technique harvesting feature is incorporated in battery [4-5]. Harvesting technique is the ideal substitution where recharging/replacing a battery is either costly or impractical. Energy harvesting (also known as energy scavenging) is the process of generating electrical energy from renewable energy sources available in environment. There exist a variety of different energy sources such as solar energy, kinetic energy thermal energy etc. Solar energy is certainly one of the most promising energy source and most of the typical applications have access to solar energy. Therefore, the energy generated by solar panels suffices to execute most common applications equipped with photovoltaic cells, due to which perpetual operation becomes possible without frequent recharging and replacement of the batteries [4].

1.1 Characteristics of energy harvesting real-time systems

1. Ability to operate by lowest standby current to maximize storage of energy.
2. Consume lowest possible energy when active.
3. Ability to turn on and off instantaneously.
4. Analog capability for sensor interfacing and measurement.
5. Ability to operate with low voltage range.
6. Lowest leakage current to maximize harvested energy.

1.2 Limitations of energy harvesting real-time systems

1. Renewable energy sources available in environment are unstable.
2. Intensity of energy from renewable energy sources varies with time, for example in case of solar energy at day time the intensity of light is very high but during night it will be zero.
3. Limited size of energy storage or battery.

Apart from all these constraints, to maximize the possible numbers of tasks to be scheduled, an efficient power management technique is required.

Present study focuses on scheduling periodic tasks with deadline, on a uniprocessor platform and variable speed system which is powered by renewable energy storage with limited capacity such as battery or a capacitor.

The content of present study is summarized in different sections. Section-2 introduces the related research works. Section-3 describes the motivational example. The energy harvesting system model and some assumptions are described in Section-4. Section-5 explains the proposed methodology with example.

Simulation results and discussions are present in Section-6. Finally Section-7 summarizes present study.

2. RELATED WORK

Energy-aware real-time scheduling is the area of intensive research in present day scenario. Most of the works in this area focuses on either minimizing the energy consumption or maximizing the system performance such as the lifetime achieved under the energy constraints. In such cases, the recharge ability of the energy storage unit is always disregarded.

Scheduling real-time tasks on a uniprocessor rechargeable system has gained a little attention. The question amounts to find a schedule which is able to execute all the tasks within the deadline and energy storage constraints i.e. without running out of energy. Moser et al. [12] focused on scheduling tasks with deadlines, periodic or not, that run on a uniprocessor system that is powered by a rechargeable storage unit.

The source power is assumed to be predictable but time-varying. They proposed lazy scheduling algorithm (LSA) and proved it to be optimal in terms of deadline miss ratio. LSA is a variation of the famous earliest deadline first scheduler, the system starts executing a task only if the task is ready and has the earliest deadline among all ready tasks and the system is able to keep on running at the maximum power until the deadline of the task.

In that work, the consumption power of the computing system is characterized by some maximum value which implies that for every task, its total energy consumption is directly connected to its execution time through the constant power of the processing device.

The main disadvantage of this work is that the LSA algorithm executes tasks at full power and therefore, future tasks will violate deadlines because of limited energy. Moreover, in practice, the total energy which can be consumed by a task is not necessarily proportional to its execution time.

Author in [6-7] presented a scheduling algorithm, EDeg (Earlier Deadline with energy guarantee) in which tasks are schedule according to an on-line algorithm that ignores the future energy production and the arrival time of tasks. It knows only the energy consumption of tasks that are released on the processor that takes into consideration the limits of both time and energy. EDeg relies on two basic concepts: slack time and slack energy. The main idea behind EDeg is to run tasks according to the earliest deadline first rule.

However, before authorizing a task to execute, we must ensure that the energy storage is sufficient to execute all future occurring tasks. When this condition is not verified, the processor has to stay idle so that the storage unit recharges as much as possible. The problem with this approach is that task are executed with a fixed speed which will some time cause energy overflow, on the other hand some time results in the insufficient energy for the task set ready to execute.

In Allavena et al. [2] describe an off-line scheduler that uses voltage and frequency selection (DVFS) for a frame based system. While they permit to reduce power consumption by slowing down task execution under deadline constraints, their approach relies on the unrealistic assumption that both the instantaneous consumption power and production power are constant.

In [10], Liu et al. propose an energy aware dynamic voltage and frequency selection algorithm, called EA-DVFS, for periodic tasks. The purpose of EA-DVFS is to efficiently use the task slack and further reduce the deadline miss rate. In this algorithm, whether or not the system slows down the task execution for energy saving depends on the available energy. If the system has sufficient energy, the task is executed at its full speed; otherwise, it is stretched and executed at a lower speed. Unfortunately, this algorithm has limited impact since, in most embedded applications, the energy storage has a non constant recharging rate and every task is characterized by its own profile of power consumption which can vary along time.

3. MOTIVATIONAL EXAMPLE

Suppose that there are three preemptive periodic tasks $\tau_1(0, 2, 10)$, $\tau_2(0, 3, 15)$, $\tau_3(1, 2, 6)$ that are required to schedule with energy and deadline constrain. And the system that we consider to schedule this set of periodic tasks is powered by a renewable energy source (eg. solar energy). The attribute of tasks τ_1 , τ_2 and τ_3 represent its release time, worst case execution time and deadline respectively.

Here we assume that the renewable energy is harvesting with a constant rate(r) during day time and which is approximately equal to 1J/sec, and the processor that we consider for scheduling the set of tasks can allow three different frequency (or) speed levels s_{low} , s_i , s_{max} . At s_{low} task consume 1J of energy per second, at s_i task consume 3J/sec and at s_{max} task consume 8J/sec. And Suppose the total capacity of energy storage (or) battery(C) = 32J, and energy available in store at $t=0$, $E_C(t) = 30J$.

At $t = 0$, there are only two task τ_1 and τ_2 is ready to execute and as the deadline of τ_1 is earlier than τ_2 . Therefore, according to EDF, priority of τ_1 is higher than τ_2 . So, at $t = 0$ τ_1 start its execution. But before that, we can check is there sufficient energy available in store to execute the task τ_1 at its maximum speed until its completion.

$$E_1(s_{max}) = 8 * 2 = 16J < 30J = E_C(t)$$

As the energy requirement of τ_1 to complete its execution at s_{max} is less than the available energy, therefore τ_1 execute at s_{max} until its completion.

At $t=1$ another task τ_3 is released and as the deadline of τ_3 is earlier than τ_1 , therefore τ_3 preempt τ_1 . Execute τ_3 , before that we will check for energy availability.

$$E_C(t) = 30 - 8 + 1 = 23J$$

And the energy requirement of τ_3 to complete its execution at s_{max} is 16J which is less than the available energy at store, therefore τ_3 execute at s_{max} until its completion, and complete its execution at $t = 3$. Therefore, at $t = 3$

$$E_C(t) = 23 - 16 + 2 = 9J$$

Here, τ_1 resume its execution, and its remaining energy requirement to complete its execution at s_{max} is 8J which is less than the available energy, therefore, τ_1 execute at s_{max} until its completion, and completed at $t = 4$. Therefore, at $t = 4$

$$E_C(t) = 9 - 8 + 1 = 2J$$

Here, τ_2 starts its execution, and as the energy requirement of τ_2

at s_{\max} is 24J, which is more than the available energy. Therefore, τ_2 execute at s_{low} . And at s_{low} , τ_2 takes 12s to complete its execution, which is more than its deadline.

4. SYSTEM MODEL AND ASSUMPTION

System consists of a uniprocessor system with a set of independent preemptive periodic tasks $\tau_1, \tau_2, \tau_3, \dots, \tau_n$. And each task τ_i has an attribute:

- a_i = arrival time of task τ_i
- $e_i(s_i)$ = worst case execution time of task τ_i at speed s_i
- $E_{\text{source}}(t_1, t_2) = (t_2 - t_1) * r \dots \dots \dots (2)$

In this energy harvesting real time system it is considered that the rate of harvesting is approximately equals to 1J/sec and is constant though out the day, and at night time the harvesting rate equals to zero.

4.2 Energy Storage

Here, we assume a limited energy storage that may be charged up to its capacity C. If no tasks are executed and the stored energy has reaches its capacity leading to energy overflow.

$$0 \leq E_C(t) \leq C \dots \dots \dots (3)$$

For executing the task, power $P_D(t)$ and the respective energy $E_D(t_1, t_2)$ is drained from the storage to execute tasks. We have the following relation:

$$E_C(t_2) \leq E_C(t_1) + E_{\text{source}}(t_1, t_2) - E_D(t_1, t_2) \quad \forall t_2 > t_1$$

Therefore,

$$E_D(t_1, t_2) \leq E_C(t_1) + E_{\text{source}}(t_1, t_2) \quad \forall t_2 > t_1 \dots \dots \dots (4)$$

4.3 Energy Drain

Energy is the function of speed level s_i . The energy drain in time interval $(t_1, t_2) E_D(t_1, t_2) = P_D(s_i)$

$E_i(s_i)$ = Energy demand of task τ_i to complete its execution with

- speed s_i
- d_i = deadline of task τ_i

Here a dynamic priority scheduling algorithm i.e, earliest deadline first scheduling algorithm is considered for assigning the priority and scheduling the set of independent periodic tasks. The DVS processor that is capable of operating at three different voltage levels V_1, V_2 and V_3 with the corresponding speed levels s_1, s_2 and s_3 is considered in this system. The speed s_1 is the lowest operating speed level measured at voltage V_1 whereas the maximum speed s_3 at the voltage level V_3 . Here the processor runs at any of the speed level between s_1 to s_3 . Power or Energy consumption at the speed s_i is given as:

$$P_i = C(s_i)^3 \dots \dots \dots (1)$$

Response time of task τ_i at speed s_i is the sum of its own execution time requirement and the execution time of its higher priority tasks preempting it.

The System modeled with energy source, energy storage, energy drain, DVS processor and real time periodic tasks as follow:

4.1 Energy Source

Harvesting source of energy is dependent on environmental factors. Such as solar, wind etc. They are highly varying with

time. Where $P_D(s_i)$ is the power drain at speed level s_i .

5. PROPOSED METHODOLOGY

In a battery-powered device, the typical power management design goal is to minimize the energy consumption (or) to maximize the lifetime achieved while meeting the required performance constraints.

In this energy aware real time scheduling, idea is to save energy by slowing down the processor just enough to meet the deadline of a task and avoid energy overflow. In this work, we proposed a harvesting aware real-time scheduling algorithm which reduce the energy as well as timing over- head by utilizing speed in such a way that response time of task is less than or just equal to the existing approach even though on the cost of lesser energy consumption. Here the execution speed of a task is selected based on the stored energy as well as available energy through harvesting and deadline of a task.

5.1 The proposed method

- Presented an approach those judiciously balance the timing as well as energy constraints.
- Here task is executed at minimum possible speed level which is just enough to meet the deadline of a task and it will be increased to next higher level if required to meet the deadline of a task or to avoid energy overflow due to recharging.
- Tasks are executed at minimum energy consuming state even though the systems have enough energy to complete the task at maximum speed.

Whenever any periodic task τ_i arrives there are two possible cases with respect of energy:

Case-1: When task requirement at maximum speed is less than available energy $E_C(t)$, then we will calculate slack time for task and if slack $(\tau_i) > 0$, reserve a time slot from latest start time of a task to deadline, and energy equals to the energy required by τ_i to complete its execution at maximum speed.

- Compute the lowest feasible speed for a task subjected to:

$$E_C(a_i) + E_H(e_i(s_i)) - E_{\text{consumption}}(\tau_i(s_i)) \leq C \dots \dots (6) \text{ and}$$

$$e_i(s_i) \leq d_i \dots \dots \dots (7)$$

- After that execute the task at assigned speed up to completion of task (or) slack time whichever is earlier.
- If task has not finished before the latest start time of a task, then execute the remaining portion of task at full speed in reserve slot with reserve energy.

Case-2: If energy requirement of task at maximum speed is greater than total available energy $E_C(t)$, then again we will calculate lowest feasible speed for a task execute the task at assigned speed until its completion and arrival of some higher priority task whichever is earlier.

5.2 Algorithm 1

Harvesting Aware Real-Time Schedule (HA-RTS):

Input: A set of n real time periodic tasks $\tau_i = \tau_1, \tau_2, \dots, \tau_n$ and a DVS based processor that support different frequency (or) speed levels.

1. Initialize a ready queue Q.
2. Initialize a reserve queue R.
3. Enter tasks ready to execute in a ready queue Q.

4. Sort the tasks in Q on EDF basis.
5. While (Q=NIL) do
6. Process the task τ_i
7. While ($E_C(t) > 0$ and $E_i(s_{\max}) \leq E_C(t)$) do
8. Calculate s_{earliest} and s_{late} of τ_i
9. If ($s_{\text{earliest}} - s_{\text{late}} > 0$), then
10. Enter task τ_i in reserve queue R
11. Reserve time slot (s_{late}, d_i) for τ_i
12. Reserve energy for τ_i , $E_{\text{reserve}}(\tau_i) = E_i(s_{\max})$
Therefore, $E_{\text{avail}}(t) = E_C(t) - E_{\text{reserve}}(\tau_i)$
13. Compute a lowest feasible speed(s_i) for τ_i using Algorithm 2.
14. Execute() task τ_i with speed s_i until its completion
15. if any other task τ_j arrive at time $a_j < f_i(\tau_i)$ and $d_j < d_i$
16. Update ready queue Q
17. τ_j preempt τ_i
18. Update $E_{\text{avail}}(t)$
 $E_{\text{avail}}(t) = E_{\text{avail}}(t) - E_{\text{consume}}(\tau_i(a_j))$
19. If ($E_j(s_{\max}) \leq E_{\text{avail}}(t)$)
20. Go to step 8., else go to step 27.
21. Else τ_i continue its execution at (s_i)
22. If τ_i completes before s_{late} , then
23. Remove τ_i from Q and R
24. Frees the reserve energy and time slot for τ_i
 $E_{\text{avail}}(t) = E_{\text{avail}}(t) + E_{\text{reserve}}(\tau_i)$
25. Else remaining portion of τ_i executed at s_{\max} and
 $E_{\text{avail}}(t) = E_{\text{avail}}(t) + (E_{\text{reserve}}(\tau_i) - E_{\text{consume}}(\tau_i))$
26. $E_{\text{avail}}(t) = E_C(t)$
en
d if
end if
end if
end
while
27. While ($E_C(t) > 0$ and $E_i(s_{\max}) > E_C(t)$) do
28. Compute a lower feasible speed (s_i) for τ_i using Algorithm 2.
29. Execute() task τ_i with speed s_i until its completion
30. While ($E_C(t) \leq 0$) do
31. Calculate common slack for all tasks ready to execute in Q at time t when $E_C(t)=0$
32. While ($E_C(t) \leq E_{\max}$ and $\text{slack}(t) > 0$)
33. Wait and recharge the battery
34. $t = t + 1$
if a_k = arrival time of task τ_k , then insert τ_k in Q and update $\text{slack}(t)$
end if
end while
end while
end while

5.3 Algorithm 2

Speed Assignment algorithm:

1. Prepare a list of all possible speed levels (or) frequency levels used in this case.

2. Sort the speed levels in their increasing order.
3. While ($s_i = s_n$) do
4. if ($E_C(a_i) + E_H(e_i(s_i)) - E_{\text{consumption}}(\tau_i(s_i)) \leq C$
and $e_i(s_i) \leq d_i$, then
5. $s_i = s_i$
6. else $s_i = s_i + 1$
7. Go to step 4.
- end if
- end while

Example: In this section we explain the above approach with the help of an example, and the effectiveness of proposed approach should get highlighted from this.

Suppose that there are three preemptive periodic tasks $\tau_1(0, 2, 10)$, $\tau_2(0, 3, 15)$, $\tau_3(1, 2, 6)$ that are required to schedule with energy and deadline constrain. And the system that we consider to schedule this set of periodic tasks is powered by a renewable energy source (eg. solar energy). The attribute of tasks τ_1 , τ_2 and τ_3 represent its release time, worst case execution time and deadline respectively.

Here we assume that the renewable energy is harvesting with a constant rate(r) during day time and which is approximately equal to 1J/sec, and the processor that we consider for scheduling the set of tasks can allow only three different frequency (or) speed levels s_{low} , s_i and s_{\max} . At s_{low} task consume 1J of energy per second, at s_i task consume 3J/sec and at s_{\max} task consume 8J/sec.

Suppose the total capacity of energy storage (or) battery, $C = 32J$, and energy available in store at $t=0$, $E_C(t) = 30J$. Therefore, at $t = 0$, there are only two task τ_1 and τ_2 is ready to execute and as the deadline of τ_1 is earlier than τ_2 . Therefore, according to EDF, priority of τ_1 is higher than τ_2 . So, at $t = 0$, τ_1 start its execution. But, before that we can check is there sufficient energy available in store to execute the task τ_1 at its maximum speed until its completion.

$$E_i(s_{\max}) = 8 \times 2 = 16J < 30J = E_C(t)$$

Therefore, we can say that there is sufficient energy available in store to execute the task τ_1 at its maximum speed until its completion, but we are not going to execute the task τ_1 at its maximum speed until there is slack available for τ_1 . Initially execute the task τ_1 at lowest possible speed until task completion (or) the slack of τ_1 becomes zero whichever is earlier.

1. Calculate $\text{slack}(\tau_1) = s_{\text{late}} - s_{\text{earliest}} = 8 - 0 = 8 > 0$
2. Since there is slack available for task τ_1 , therefore we reserve the time slot from (s_{late} to d_1) and energy equals to 16J for τ_1 and enter the τ_1 in reserve queue R. After reserving energy for τ_1 we can assume that only 14J of energy is available for executing all the ready task in ready queue Q.

$$E_{\text{avail}}(t) = E_C(t) - E_{\text{reserve}}(\tau_1) = (30 - 16) = 14J$$

3. Calculate a lowest feasible speed for τ_1

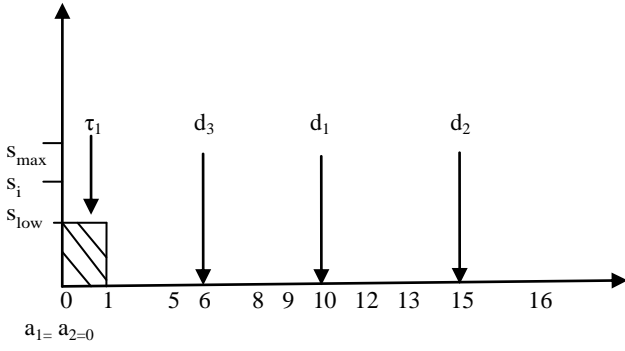


Figure1. Schedule with deadline and energy constraint 1.

Now, start from lowest speed level s_{low} and check whether it is feasible to execute the τ_1 at this speed level (or) not. checking the feasibility of τ_1 at s_{low} and energy consumption

$$e_1(s_{low}) = 8\text{sec} < d_1$$

$$E_{consume}(\tau_1(s_{low})) = 8 * 1 = 8J < 14J = E_{avail}(t)$$

Also, Total available energy in store, $E_C(t) = 30 + 8 - 8 = 30J < C$ (checking for energy overflow)

As both the condition is true i.e., at this speed task τ_1 completes its execution before its deadline also no energy overflow will

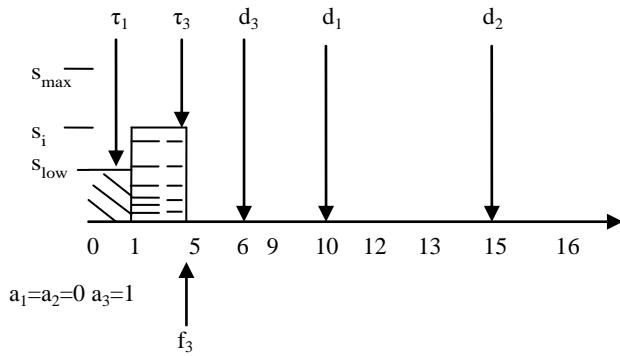


Figure2. Schedule with deadline and energy constraint 2.

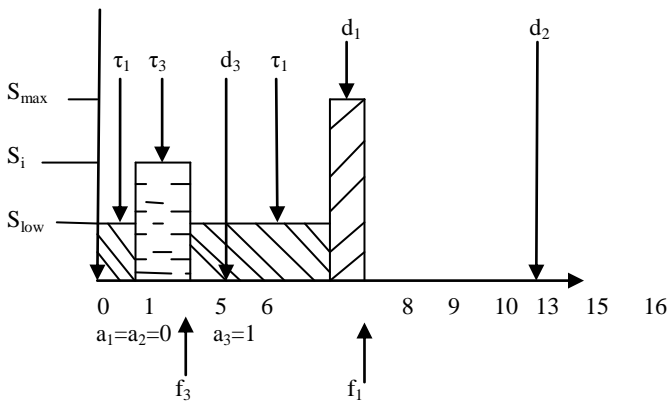


Figure3. Schedule with deadline and energy constraint 3.

occur. So, we can execute the task τ_1 with the lowest speed s_{low} .

4. Now execute the task τ_1 at s_{low} until its completion (or) the arrival of some higher priority task whichever is earlier till $t = 8 = s_{late}$ (or) until slack = 0.

After that if, τ_1 remains incomplete, and then the remaining portion of τ_1 is executed at maximum speed (s_{max}) until d_1 i.e., $t=10$ (or) its completion whichever is earlier.

1. At $t=1$ another task τ_3 is released and as the deadline of τ_3 is earlier than τ_1 , therefore τ_3 preempt τ_1 .
2. Execute τ_3 , before check for energy availability.

$$E_{avail}(t) = 14 + 1 - 1 = 14J \text{ and}$$

$$E_3(s_{max}) = 16J > 14J = E_{avail}(t)$$

Here, the energy requirement of a task is more than the available.

3. Calculate the lowest feasible speed for τ_3 , if τ_3 execute at speed s_{low} , then time taken by τ_3 to complete its execution, $e_3(s_{low}) = 8$ sec and τ_3 released at $t=1$. Therefore, $f_3 = 8 + 1 = 9 > d_3 = 6$

So, if execute τ_3 with the speed s_{low} it will miss its deadline. Therefore s_{low} is not the feasible speed for τ_3 .

Now Will try the next higher speed level i.e., s_i , if τ_3 execute with speed s_i , then

$e_3(s_i) = 4$ sec and therefore, $f_3 = 5 < d_3 = 6$, so at this speed level τ_3 will not miss its deadline.

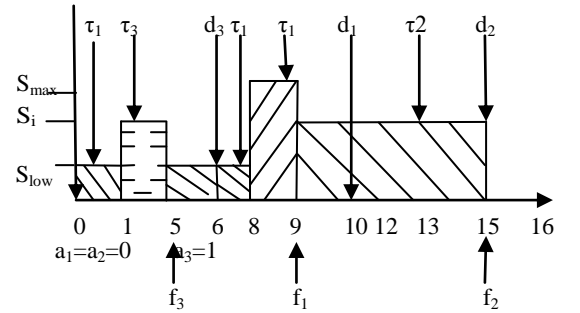


Figure4. Schedule with deadline and energy constraint 4.

requirement of τ_2 at its maximum speed $E_2(s_{max}) = 24J > E_C(t)$.

Therefore, select a lowest feasible speed for τ_2

3. Calculate lowest feasible speed for τ_2 . If τ_2 execute at speed level s_{low} , then time taken by τ_2 to complete its execution, $e_2(s_{low}) = 12$ sec and as τ_2 started its execution at $t=9$, therefore, $f_3 = 9 + 12 = 21 > d_2 = 15$.

So, if we execute τ_2 at s_{low} it will miss its deadline. Therefore, go for next higher speed level i.e s_i .

Now, the time taken by τ_2 to complete its execution a

Now, Total available energy in store, $E_C(t) = 30 + 4 - 12 = 22J < C$, also no energy overflow occur at this speed level. Therefore, we can say that τ_3 can execute feasibly with the speed s_i .

1. At $t=5$ τ_3 completes its execution, therefore $E_{avail}(t)=14 + 4 - 12=6J$

2. Now here at this point τ_1 resume its execution and continue its execution with the speed s_{low} up to $t=8$. Therefore, at $t = 8$

1. $E_{avail}(t) = 6 + 3 - 3 = 6J$

2. Now at this point if τ_1 remain incomplete, then the remaining portion τ_1 is execute at maximum speed (s_{max}) with reserved energy.

3. At its maximum speed τ_1 completes its execution at $t=9$. Therefore, at $t = 9$

1. $E_{avail}(t) = 6 + 1 + (16 - 8) = 15J = E_C(t)$

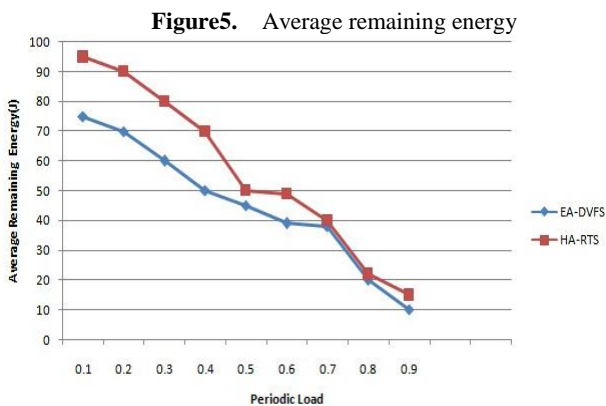
2. Now at this point τ_2 start its execution and as energy
 $s_1, e_2(s_1) = 6\text{sec}$ and $f_3 = 9 + 6 = 15 = d_2$ also,
 $E_C(t) = 15 + 6 - 18 = 3J$

Therefore, execute τ_2 at speed s_1 , so that it can feasibly schedule before its deadline without energy overflow.

Now by the above scheduling scheme we had feasibly schedule the tasks τ_1 , τ_2 and τ_3 with energy and timing constrain, also without causing any energy wastage, and at the end we have left with 3J of energy that can be utilize to further schedule tasks coming in future.

6. EXPERIMENTAL RESULTS AND DISCUSSION

Here, we compare the performance of proposed approach Harvesting Aware Real-Time Scheduling(HA-RTS)algorithm with existing approach Energy Aware Dynamic Voltage and Frequency Selection(EA-DVFS)[16]. The key parameters for performance measurement are remaining energy and tasks acceptance ratio. In the following section we measure the effect of variation in periodic tasks load on the average energy consumption and acceptance ratio of task set. The effect of load on the remaining energy consumption can be seen from the figure 6.

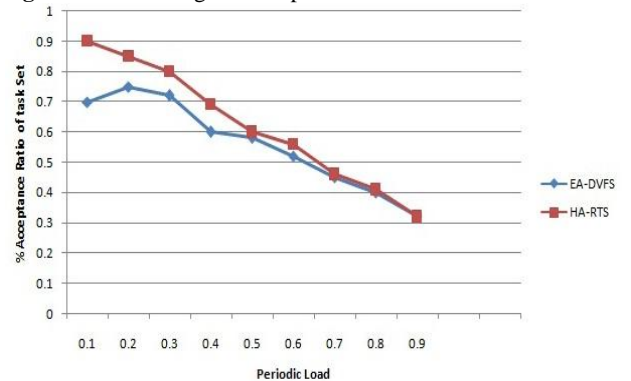


The storage capacity is to 2000J. We observe from the figure 6 when the periodic tasks load increases the remaining energy will decreases. When we varies periodic load from 10%to 90% we observe from the figure as load increase remaining energy of the system decreases. At lower periodic load (10% to 40%) our proposed approach have significant saving in energy almost store 20% more energy as compared to existing approach. This is due to, at lower periodic load our approach run the whole computation of task most of the time at slower speed and same speed level however, existing approach execute some portion at

lower speed and remaining computation time at maximum speed level even there is no any higher priority tasks. Energy consumption of task at maximum speed level increases exponentially as compare to energy consumption at lower speed level. However, at higher periodic load (70% to 90%) our proposed approach has slightly saving in energy. Most of the time our proposed approach as well as existing approach, execute the task at maximum speed level at higher periodic load. This is due to that at higher periodic load in both approaches processor rarely has chance to slowdown the speed of task.

Effect of load on acceptance ratio: The effect of load on the acceptance ratio of set of periodic tasks can be seen from the figure 7, compare the performance of existing approach and proposed approach. In this we set the storage capacity is to 2000J. We observe from the figure7 when the tasks load increases the acceptance ratio will decreases. At lower periodic load (10% to 40%) our proposed approach have accept 10% more periodic tasks as compared to existing approach. This is due to that at lower tasks load our approach run the whole computation of task most of the time at slower speed and execute whole computation at same speed level however, existing approach execute some portion at lower speed and remaining computation time at maximum speed level even though there is no any other tasks. Thus, the future arriving task may miss their deadline due to the shortage

Figure6. Percentage of acceptance ratio of task set



of energy. However, at higher tasks load (70% to 90%) our proposed approach and existing approach both accept almost same number of periodic tasks. This is due to at higher tasks load both approaches execute task at maximum speed level most of the time.

Reduction in rejection ratio: The effect of storage capacity on the rejection ratio of periodic tasks can be seen from the above figures. Our objective is concerning on the relative capacity savings achieved with our algorithms, we are especially interested in the smallest capacities C necessary to avoid any deadline violations due to the shortage of energy. We can observe from the figure when periodic load is 20%, 40%, 60% and 80% the proposed approach significantly reduces almost 50%, 30%, 20% and 10% on average respectively. We can also observe that the storage capacities saving are 48%, 40%, 10% and 5% at periodic load 20%, 40%, 60% and 80% respectively.

7. CONCLUSION

Here, a general scheduling algorithm that maximizes the utility of harvested energy for real time embedded system with voltage scalable processor is presented. The proposed approach judiciously decides operating speed that reduce the energy overhead as well as timing overhead due to the speed switching.

The examples and simulation studies shows that the proposed scheduling algorithm improves the overall average remaining stored energy. The average remaining stored energy of the system is approximately 5% more than the existing approach when a load varied from 70% to 90% while 20% more energy will be stored at lower tasks load varied from 10% to 50%. When the tasks load is low say 10% to 50% our proposed approach accepts 8% more task than existing approach. However, at higher tasks load both approach Perform almost same.

Thus, extensive simulation and illustrative example shows that our proposed approach is capable of performing better

in terms of average stored remaining energy of the system as well as acceptance ratio of periodic tasks.

8. FUTURE SCOPE

In present study, the problem of energy minimization was solved for periodic load. So, we can extend this work for aperiodic and sporadic load along with periodic load.

9. REFERENCES

- [1] Agrawal, S., Yadav, R. S. and Ranvijay, 2009. A Pre-emption Control Approach for Energy Aware Fault Tolerant Real Time System, *International Journal of Recent Trends in Engineering*, 381-386.
- [2] Allavena, A. and Mosse, D., 2001. Scheduling of frame-based embedded systems with rechargeable batteries, *Workshop Power Manage Real-time Embedded System*.
- [3] Bertogna, M. and Baruah, S., 2010. Limited Preemption EDF Scheduling of Sporadic Task Systems, *IEEE Transactions on Industrial Informatics*, 579 - 591.
- [4] Chetto, M. and Zhang, H., 2010. Performance Evaluation of Real- Time Scheduling Heuristics for Energy Harvesting Systems, *EEE/ACM Int'l Conference on Cyber, Physical and Social Computing (CPSCoM), Green Computing and Communications (GreenCom)*, 398 - 403.
- [5] Dehghan and Maryam, 2010. Adaptive checkpoint placement in energy harvesting real-time systems , 18th Iranian Conference on Electrical Engineering (ICEE), 932 - 937.
- [6] Hussein, E. L. G., Chetto, M. and Chehade, R.H., 2011. EH-EDF: An On-line Scheduler for Real-Time Energy Harvesting Systems, 18th IEEE International Conference on Electronics Circuit and System (ICECS), 776-779.
- [7] Hussein, E. L. G., Chetto, M. and Chehade, R.H., 2011. A real-time scheduling framework for embedded systems with environmental energy harvesting, *Elsevier on Computers and Electrical Engineering* 37.
- [8] Liu, S., Lu, J., Wu, Q. and Qiu, Q., 2011. Harvesting-Aware Power Management for Real-Time Systems With Renewable Energy, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 1 - 14.
- [9] Liu, S., Lu, J., Wu, Q. and Qiu, Q., 2010. Load-matching adaptive task scheduling for energy efficiency in energy harvesting real-time embedded systems, *ACM/IEEE International Symposium on Low-Power Electronics and Design (ISLPED)*, 325 – 330.
- [10] Liu, S., Qiu, Q. and Wu, Q., 2008. Energy Aware Dynamic Voltage and Frequency Selection for Real-Time Systems with Energy Harvesting, *conference on Design, Automation and Test in Europe*, 236 - 241.
- [11] Lu, J. and Qiu, Q. 2011. Scheduling and mapping of periodic tasks on multi-core embedded systems with energy harvesting, *Journal, Computers and Electrical Engineering archive*, 498 - 510.
- [12] Moser, C., Brunelli, D., Thiele, L. and Benini, L., 2007. Real-time scheduling for energy harvesting sensor nodes, *Real-Time Syst ; 37(3)*, 233-260.
- [13] Moser, C., Thiele, L., Brunelli, D. and Benini, L., 2007. Adaptive Power Management in Energy Harvesting Systems , *Design, Automation and Test in Europe Conference and Exhibition*, 1 - 6.
- [14] Niu, L. and Quan, G. 2006. System-Wide Dynamic Power Management for Portable Multimedia Devices, *Eighth IEEE International Symposium on Multimedia*, 97 -104 .
- [15] Niu, L. and Quan, G., 2006. Energy minimization for real-time systems with (m,k)-guarantee, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 717-729.
- [16] Paul, A. A. and Pillai, A.S. B., 2011. Reducing the Number of Context Switches in Real Time Systems, *International Conference on Process Automation, Control and Computing (PACC)*, 1 - 6.
- [17] Qadi, A., Goddard, S. and Farritor, S., 2003. A dynamic voltage scaling algorithm for sporadic tasks , 24th IEEE Conference on Real- Time Systems Symposium (RTSS), 52 - 62.
- [18] Zhu, L., Tongquan, Wei, T., Yonghe, Guo., Xiaodao, Chen. and Shiyan, Hu., 2010. Energy efficient fault-tolerance task allocation scheme for real-time energy harvesting systems, *International Conference on Intelligent Control and Information Processing (ICICIP)*, 589 - 594.