The CSI Journal on
Computer Science and Engineering
Vol. 12, No. 2 & 4 (b), 2014
Pages 36-46
Regular Paper

Temperature-Aware Speed Scheduling in Periodic Real-Time Systems

Morteza Mohaqeqi[1] Mehdi Kargahi[1,2] Fatemeh Gharedaghi[1]

[1]School of Electrical and Computer Engineering, University of Tehran, Tehran, Iran
[2]School of Computer Science, Institute for Research in Fundamental Sciences (IPM), Tehran, Iran

Abstract

Real-time embedded systems usually work in harsh environments with variable ambient temperature. As temperature has a significant impact on the processor reliability, thermal management is mandatory in many such systems, where it is critical to keep the processor temperature below some specified threshold. To achieve this goal, however, the system performance may adversely be affected. Our favorite performance objective in this paper is the minimization of the fraction of jobs which miss their deadline, regarding the thermal constraint. We propose a quasi-static approach based on dynamic voltage scaling: it first generates a number of thermally feasible speed schedules; one schedule for each range of ambient temperatures; then, at runtime, the appropriate speed schedule is selected according to the ambient temperature. An online algorithm is also proposed to decide when the switching between the schedules occurs. The latter algorithm brings the processor temperature below specific values, so the newly selected speed schedule starts to work at the new scheduling without violating the temperature constraint. The efficiencies of the proposed methods are investigated by simulation experiments.

Keywords: Dynamic Voltage Scaling (DVS), Real-Time Systems, Thermal Constrained Scheduling, Variable Ambient Temperature.

1. Introduction

Evolutions in the semi-conductor technology and feature size have led to increasing power density in modern microprocessors. As an effect, overheating has become a major problem in many computing systems because of its significant impacts on the system reliability, performance, and power consumption [1]. This problem gets more serious in embedded systems, especially those working in harsh environments with highly variable temperature. Employing cooling devices (like heat sink or fan) inside the systems is restricted due to the energy, cost and size limitations [2].

High temperature can lead to degraded reliability and performance [1]. Recent studies [3] show that overheating is the cause of more than 50% of electronic failures. Timing errors are the other impact of temperature violation [4]. Accordingly, temperature management becomes more prominent in the design of real-time embedded systems. The management can be performed at the design time through advanced packaging and cooling solutions, and at run-time, through various dynamic thermal management (DTM) techniques [5].

Several hardware and software DTM techniques have been proposed to control the system power consumption and/or reliability. Hardware-based techniques such as clock-gating [6], fetch-gating [7] and dynamic voltage scaling (DVS) [8] are employed to reduce power consumption. Besides, other system-level techniques such as temperature-aware scheduling [9, 10, 11, 12] and task migration [13] try to distribute and balance the power to better guarantee the temperature constraints.

In real-time systems, however, temperature-aware scheduling is more challenging. Dynamic power management (DPM) and DVS are usual techniques in such systems to reach the performance goals without violating the thermal constraints [14]. DPM-based methods [15, 16] use idle times to reduce temperature by putting the processor into

idle modes. As another approach, DVS is used as a major approach to control the thermal behavior of the systems to avoid processor overheating through dynamic speed scaling [10, 11] (named as speed scheduling). However, this approach may impose some undesired performance degradation, especially in real-time systems where each job has a time constraint (deadline) that its violation negatively impacts the overall system performance. From the thermal perspective, the ambient temperature may change, resulting in situations where some adaptations in the schedule are required to allow the system to continue its operation with an appropriate performance.

In this paper, we focus on a firm real-time (FRT) system containing a DVS-enabled processor and variable ambient temperature. In a FRT system, jobs violating their deadline are of no value and should be thrown away of the system. The goal of this paper is to minimize the system deadline miss ratio while the it is protected against overheating.

We present a temperature-aware speed scheduling algorithm to determine the speed level of the processor for each job, aiming at minimizing the number of missed jobs. The variability of temperature in unpredictable physical environments is taken into account by proposing a quasi-static scheduling method. In the scheme, an offline algorithm is proposed to find a set of speed schedules for different ranges of ambient temperature. At runtime, whenever an ambient temperature change occurs, the online part of the method chooses another schedule which is more appropriate for the new condition. Another online algorithm is also proposed to handle transitions between the schedules, when the ambient temperature changes in the middle of the current schedule (unlike at the schedule boundaries).

In summary, the major contributions of this paper are as follows: (1) We propose an optimal speed scheduling algorithm for a set of FRT jobs which minimizes the number of jobs that miss their deadline while satisfying an specified temperature constraint in one hyper-period; (2) based on this optimal algorithm, we propose a heuristic method for finding a speed schedule that satisfies the thermal constraint in the steady-state and exhibits small number of missed jobs; (3) a quasi-static method is also proposed to appropriately adapt the system with the environment temperature variations through proposing a low overhead algorithm for runtime switching between different speed schedules.

The remainder of this paper is structured as follows. There is a brief survey on the recent thermal management techniques with a concentration on real-time systems in Section 2. Section 3 provides the system model, some required basic information, and the problem definition. Offline phase of our scheme is presented in Section 4. Based on this scheme, Section 5 presents the quasi-static method for speed scheduling in a FRT system with variable ambient temperature. Experimental results are presented in Section 6 and finally the paper is concluded in Section 7.

2. Related Work

Different DVS and DPM techniques have been proposed as solutions to dynamic thermal management (DTM) [17]. DTM techniques are generally divided into two categories: reactive and proactive. The reactive techniques are triggered when the temperature exceeds a specified threshold; while, according to proactive techniques, system responds to thermal emergencies before they occur.

Brooks et al. [7] have evaluated the benefits of several reactive hardware-based DTM techniques such as fetch-gating that stall fetching instruction and use the instruction fetch window to feed the pipeline. These hardware-based techniques change the architecture to reduce power dissipation and chip temperature.

DVS has been applied in many real-time systems with the aim of reactive or proactive scheduling for thermal management [11, 18, 5, 12, 19]. Wang and Bettati [11, 18, 5] have proposed reactive two-speed schedule for hard real-time periodic tasks under thermal constraint. They have considered a processor with two speed levels which runs at the highest speed until the temperature reaches to its maximum acceptable value. At this time, the processor enters to the equilibrium mode and run with equilibrium speed. Working with this speed, the temperature never exceeds the specified threshold. Since modern processors support several speed levels, this approach would be more efficient if it employs more levels of speed.

Another reactive speed scheduling for a hard real-time system is presented in [20] with the aim of minimizing peak temperature. It is called m-oscillating algorithm. This algorithm first considers two-speed schedule and then divides each high and low interval evenly into m sections, and run the processor with the low and high speeds alternatively. Also, it is shown that the m-oscillation scheme is very effective in reducing the peak temperature than reactive two-speed scheme.

On the other hand, proactive methods are usually offline [12, 19]. In [12], a proactive speed scheduling has been developed for real-time tasks under thermal and timing constraint. To obtain a feasible speed schedule, two approaches are proposed in the paper; timing-optimization and thermal-optimization. The first approach aims to minimize the delay in completing tasks without violating thermal constraints and the second approach aims to minimize the temperature at the beginning of the period. Also, Wang et al. [19] have presented a proactive energy-efficient speed scheduling for periodic real-time tasks which minimizes energy consumption under thermal and timing constraints. Note that both [12] and [19] assume that the processor supports continuous range of speeds, but as an extension they present an approximation solution for discrete speed levels too.

Some methods address thermal-aware design problem as a performance optimization problem under temperature limits. Zhang and Chatha [17] have solved this problem to minimize total execution time under thermal constraints. They have assumed a periodic sequence of jobs, and discrete levels of processor speed. The goal is to assign a speed level to a job, and select the processor sleep times such that the total execution time of all the jobs is minimized under the temperature constraint [17]. To solve this problem, they have presented a pseudo-polynomial optimal algorithm based on dynamic programming approach. We apply this approach and extend their idea to solve our problem.

In the context of real-time systems, there have been a few works that use DPM to consider real-time constraints under thermal limitations. Thiele et al. [15] present a DTM policy for a single speed processor that at any time choose the mode of processor (active or idle) such that the jobs meet their

deadlines and the peak temperature is minimized. In this approach, the execution times of jobs are delayed by distributing the idle times between the executions. Another DPM approach is presented in [16], where the processor is periodically put in the idle mode for a specific duration to prevent thermal violation. At higher environment temperature, their method increases the idle intervals. However, in order to meet all the deadlines, they select a lower real-time performance mode, in which the total number of jobs that should be run is less than the previous mode.

There are also other studies about schedulability analysis of real-time periodic tasks under temperature constraint. Quan et al. [21] have shown that a schedule which satisfies the thermal constraint within the first hyper-period, is not necessarily feasible in the steady state. Also, they have addressed the temperature/leakage relation in the feasibility analysis. These studies are important since they emphasize on the conditions of steady state schedulability. We will apply their technique to show the temperature feasibility of our schedule in the steady state.

In [17], Zhang and Chatha guarantee the temperature feasibility of periodic job sets by forcing the temperature at the end of one complete execution of the job set to be equal or less than beginning of the period.

In [22], Quan et al. first study the problem to guarantee both the timing and the temperature constraints for a periodic hard real-time task set, scheduled according to the EDF policy. Then, they propose a speed scheduling algorithm under thermal constraints based on a schedulability analysis. Their algorithm is developed according to an optimal energy-saving scheduling algorithm [22]. They assume that there is an energy-aware speed scheduling, and then, update it based on the temperature constraint.

In what follows, we develop a method for proactive speed scheduling of periodic FRT systems under the maximum temperature constraint to minimize the missed jobs. Then in Section 5, we apply the derived speed schedules in a quasi-static approach for the case of variable ambient temperature.

3. System Model and Problem Definition

In this section, we explain the processor model including its thermal behavior, the job model, and some necessary definitions and lemmas including the problem definition.

3.1. Processor Model

The processor has K active speed modes s_k, $1 \le k \le K$ (and equivalently, K voltage levels v_k) and one sleep mode s_{sleep}. The processor can dynamically switch between the modes, namely it has the DVS capability. For this processor, we assume that the power consumption in an active mode s_k, $1 \le k \le K$ is a convex function of the speed, i.e. $P(s_k) \simeq {s_k}^{\alpha}$, $\alpha > 1$, where the processor speed is proportional to the respective voltage level, namely $s_k \propto v_k$ [18]. In this paper, similar to [6, 14], we assume $\alpha = 3$.

The thermal model of the processor is introduced based on the lumped RC thermal model of Skadron et al. [20], i.e. based on the existing duality between heat transfer and electricity transfer in RC circuits. Let $P(t)$ and $T(t)$, respectively, denote the power consumption and temperature of the processor at time t. Assuming a fixed ambient temperature T_{amb}, then the dynamic behavior of the temperature is given by:

$$RC\,\frac{dT(t)}{dt} + T(t) - RP(t) = T_{amb}, \quad (1)$$

where R and C are respectively the thermal resistance $(J/°C)$ and capacitance $(W/°C)$. According to this model, if the processor works in mode s_k during time period $[t_0, t]$ with an initial temperature $T(t_0)$ at time t_0, the final temperature at time t is given by:

$$T(t) = T_s + (T(t_0) - T_s)e^{-b(t-t_0)}. \quad (2)$$

where $b = \frac{1}{RC}$, T_s is the steady-state temperature, and is calculated as $T_s = P_{s_k}R + T_{amb}$ and P_{s_k} is power consumption in mode s_k. To protect the processor from overheating, a threshold is defined for the temperature, denoted by T_{max}. Due to the reliability purposes, the processor temperature should not violate T_{max}.

3.2. Job Model

We consider independent periodic FRT tasks, which are scheduled based on an arbitrary non-preemptive scheduling policy. The least common multiplier of all the task periods is the hyper-period, denoted by H. It contains the set of n jobs $J = \{J_1, J_2, \dots, J_n\}$, where n is calculated as the sum of the number of instances of all the periodic tasks in the hyper-period. These jobs are exactly repeated in the next hyper-periods. Each job J_i is represented as (r_i, c_i, D_i), where r_i and D_i are the job release-time and relative deadline, respectively. c_i, is the execution time of J_i at the maximum speed s_K.

3.3. Thermally Feasible Speed Schedule and Problem Definition

In the considered system, a speed schedule S within interval $[0, H]$ determines one speed mode for each job. Assigning zero as the speed of a job in a speed schedule means that this job will not be executed at all, and consequently, it will miss its deadline. This situation occurs if some jobs are to be thrown away due to the temperature constraints in the system. More formally, S is defined as a set of (J_i, s_k), where the pair determines the speed mode s_k for job J_i.

In this paper, we consider thermally feasible speed scheduling of a set of periodic FRT jobs according to the following definitions:

Definition 1. A speed schedule S is said to be locally thermally feasible (LTF) if for a given initial temperature T_{init} as the temperature at start of the hyper-period, the processor temperature never exceeds T_{max} along with that hyper-period.

Definition 2. A speed schedule S is said to be thermally feasible in the steady state (TFSS) [21] if when the system which is working under this speed schedule reaches to steady state, it never violates the temperature constraint T_{max}. To determine whether a speed schedule is TFSS or not, Quan et al. [21] has proposed necessary and sufficient conditions as follows:

Lemma 1 (Sufficient Condition). If under a LTF speed

schedule S, temperature T_H at the end of the first hyper-period is less than or equal to the initial temperature T_0 of the hyper-period, i.e. $T_H \leq T_0$, then it is also globally feasible (namely, it is TFSS).

Proof: See [21].

Lemma 2 (Necessary and Sufficient Condition). A speed schedule S is TFSS if and only if for all time instants $t_m \in [0, H]$, the following condition holds:

$$T(t_m) \leq T_{max} - \left(\frac{T_H - T_0}{1 - e^{-bH}}\right) e^{-bt_m^{\sim}} \quad (3)$$

Proof: See [21].

Based on above descriptions, our speed scheduling problem is defined as follows:

Problem 1. Given an ambient temperature T_{amb}, an initial temperature T_{init}, and a threshold temperature T_{max}, the problem is to find a TFSS speed schedule for a set of FRT jobs J in a hyper-period of length H which are scheduled by a non-preemptive scheduling policy on a DVS-enabled processor, that has the minimum number of missed jobs.

4. Speed Scheduling Under Thermal Constraint

Our approach to solve Problem 1 consists of two steps. In the first step, we solve the problem for the first hyper-period. More precisely, for each number of missed jobs, a speed schedule will be found (if it exists) that is LTF in the first hyper-period and has some desired properties. The result of the first step is a set of LTF schedules. In the second step, a speed schedule that has the least number of misses and is TFSS will be selected from this set as the target schedule.

Section 6.1 describes an optimal algorithm for the first step, employing a dynamic programming (DP) technique. By optimality, we mean that if for a given ordered set of jobs and a minimum number of missed jobs m there exists a LTF schedule, then this algorithm will certainly find it. Note that such speed schedule may not be unique and our algorithm doesn't find all of them. The second step, which is specified in Section 6.2, utilizes the results to derive an appropriate solution for Problem1.

4.1. Optimal Speed Scheduling for the First Hyper-Period

We present an optimal speed scheduling algorithm based on dynamic programming (DP) approach to find LTF schedules for the first hyper-period. This algorithm, which is called minimum final temperature (MFT) only considers the first hyper-period and for each number of missed jobs in the first hyper-period, finds a thermally feasible speed schedule that has minimum temperature at the end of that hyper-period. The reason that we are interested in low final temperature is that if a speed schedule has a quite low final temperature, it is more likely to satisfy sufficient condition of being TFSS (Lemma 1) which is the aim of the final algorithm.

MFT is inspired from Zhang [17] where a dynamic programming algorithm is introduced for scheduling non-preemptive non-real-time jobs. We will extend their idea to support jobs with timing constraints.

In order to optimally solve the described problem, we propose a dynamic programming algorithm with a 3-dimension table. As defined in Section 3, the processor can operate in either sleep mode or any of K active modes. We integrate sleep mode in our solution by considering sleep jobs between active jobs. Thus, we have $n' = 2n$ jobs in the job set $J = \{J_1, J_2, \dots, J_{2n}\}$. When i is odd, J_i refers to actual job $J_{(i+1)/2}$ and when i is even, it refers to a sleep job. For sleep jobs, we consider K' executing mode, $\{d_1, \dots, d_{K'}\}$, determining the length of sleep job. The length of sleep jobs specify the sleep time of processor. Speed scheduling for sleep jobs is equivalent to selecting one of these K' modes. Note that actual jobs are the only jobs subjected to deadline misses.

In our dynamic programming approach, we define $T(i, Z, m)$ as minimum final temperature where i jobs are executed in exactly $Z \in \{1,2, \dots, H\}$ time units such that $m \in \{0,1,2, \dots, n\}$ jobs of i jobs are missed. In other words $i - m$ jobs are executed in exactly Z time units. We specify a recursive relation to find $T(i, Z, m)$. To this aim, we should consider two situations: i is even, and i is odd.

If i is odd, then there exist two cases: the last job (J_i) is missed or it is executed. In the first case, it is assumed that the last job J_i is missed and there is a solution which gives us a temperature below T_{max} with $m - 1$ misses ending to time Z. We denote this temperature value by T_{miss}. According to second case, there are K choices for selecting the speed level for job J_i. However, for each speed level that can be selected for execution of this job, we should verify that the deadline of the job is after Z and the job is released before $Z - c_{ik}$, where c_{ik} is the execution time of J_i in speed mode s_k. It gives us a set of temperature values (named as T_{exe}) with at most K elements. For each element of the set (which is related to one of speed levels, s_k), we know the minimum possible temperature to execute $i - 1$ jobs with m misses in $Z - c_{ik}$ unit of time, which is determined in step $i - 1$. We can also easily calculate the temperature rise (or fall) due to execution of i th job, J_i, in c_{ik} unit of time (with speed s_k). Then the final temperature is obtained by adding up these two values. Based on these statements, we define the following recursive relation for $T(i, Z, m)$ where i is odd.

$$\begin{aligned} T_{exe} = \{&T(i-1, Z - c_{ik}, m) + \Delta T(s_k) | \\ &r_i \leq Z - c_{ik}, Z \leq D_i, T(i-1, Z - c_{ik}, m) \neq \infty, k \in [1, K]\} \\ T_{miss} = \{&T(i-1, Z, m-1) \mid r_i \leq Z\} \\ T(i, Z, m) = \text{Min}\{&t | t \in (T_{exe} \cup T_{miss}), t < T_{max}\} \end{aligned} \quad (4)$$

In the above formulation, $\Delta T(s_k)$ shows temperature rise (or fall) due to execution of job J_i in speed mode s_k. If no feasible solution with available speed mode exists which ends in Z time unit, then we set $T(i, Z, m) = \infty$. We also define $S_{i,Z,m}$ as a speed schedule which results in $T(i, Z, m)$.

If i is even, T_{miss} is considered as an empty set, because when we want to schedule sleep jobs, we do not change the number of misses so far. Consequently, to attain $T(i, Z, m)$ for sleep jobs, we have:

$$\begin{aligned} T_{exe} = \{&T(i-1, Z - d_l, m) + \Delta T(d_l) | \\ &T(i-1, Z - d_l, m) \neq \infty, l \in [1, K']\} \\ T(i, Z, m) = \text{Min}\{&t | t \in T_{exe}, t < T_{max}\} \end{aligned} \quad (5)$$

```
/* initialization */
1.  for Z = 0 to H
2.  T(0, Z, 0) = T_init;

3.  fori = 1 to n'
4.      forZ = 0 to H
5.          if(T_exe == ∅)
6.              T(i, Z, 0) = ∞
7.          else
8.              T(i, Z, 0) = min {t ∈ T_exe}
/* filling the table */
9.  fori = 1 to n'
10.     forZ = 0 to H
11.         form = 1 to n
12.             T(i, Z, m)= min {t∈T_exe∪ T_miss}
```

Figure 1. Construction of table T

The recursive formula (4) and (5) provide a basis for construction of a 3-dimension table where every cell (i, Z, m) corresponds to the minimum final temperature when execution of i jobs is considered in Z time unit and m jobs of i jobs are missed. Figure 1 shows the incremental construction of the 3-dimention table. As initial values, first T(i, Z, 0) is filled for i = 1 to 2n according to (4), where for each value of i, Z varies from 1 to H. Then the other cells are calculated row by row.

Note that for those cells which have not any value, or are not schedulable, T(i, Z, 0) = ∞. Also a frequency table which is named F with the same size of temperature table T is constructed to track the speed mode for each job. Each cell in the frequency table is filled with the speed mode that leads to the respective temperature at the same cell in the T table. Therefore, the relative speed schedule for each number of miss can be achieved by tracking back in the F table from n' to1.

After filling all cells of table T, for each m, T(n′, H, m) indicates the minimum final temperature that can be achieved by missing m jobs under temperature constraint. Using frequency table F, we can find the final speed schedule S_{n',H,m^*}, which is ended to the cell T(n′, H, m*) in temperature table. This speed schedule determines the LTF speed schedule with minimum missed jobs where,

$$m^* = \arg \mathrm{Min}_{0\le m\le n}\{T(n', H, m) < \tilde{T_{max}}\} \quad (6)$$

Proof of optimality: Based on the recursive relation (4), each job can be executed or missed. Suppose it is executed with one of the K levels of speed modes s_k. According to thermal equation (2), the lower initial temperature leads to lower final temperature at the end of the execution. Therefore, T(i, Z, m) is minimum when we have minimum temperature at the initial point T(i − 1, Z− c_{ik}, m). This value is the minimum temperature when i − 1 jobs are executed in Z− c_{ik} time unit such that m jobs of i jobs are missed. So T(i, Z, m) which is the sum of T(i − 1, Z− c_{ik}, m) and ΔT(s_k), is also minimum. If the job is missed the initial point is equal to T(i − 1, Z, m − 1) which is also minimum. Finally each of them (execution or miss) that leads to minimum temperature will be selected as the optimal solution.

4.2. Speed Schedule for Steady State

For each number of misses, the MFT algorithm presents a LTF speed schedule with minimum final temperature. As mentioned, a LTF speed schedule is not necessarily TFSS. Therefore it is needed to check the derived LTF schedules in order to find one that is TFSS and has minimum number of missed jobs. The algorithm to achieve a TFSS speed schedule (named as TFSS_SS) is summarized in figure 2 Let S_{n',H,m^*} be the LTF speed schedule with m* minimum number of missed jobs. As described in subsection 3.3, Lemma 2 provides the necessary and sufficient condition to predict if a speed schedule which is feasible in the first hyper-period, is also feasible in steady state or not.

Input: J: Job set, T_{amb}: ambient temperature,
T_{max}: temperature threshold, T_{init}: initial temperature.
Output: TFSS speed schedule.
TFSS_SS ($J, T_{init}, T_{amb}, T_{Max}$)
1. MFT($J, T_{init}, T_{amb}, T_{Max}$); //both temperature table and frequency table ($T$and$F$) are constructed
2. chk = false;
3. **while** (chk = false)
4. S^*=S_{n',H,m^*}; // it is derived from F table
5. chk = check_feasibility (S^*); // based on Lemma 2 check $S^*$is feasible in the steady state or not
6. **if** (chk is true)
7. break;
8. **else**
9. $m^* = m^* + 1$;
10. **return**S^*

Figure 2. TFSS_SS algorithm

Based on Lemma 2, if speed schedule S_{n',H,m^*} is feasible in the steady state the final solution has been found and the procedure is terminated. Otherwise we should check the feasibility for the speed schedule with m* + 1 number of missed jobs. This procedure is repeated until the TFSS speed schedule is found. It should be noted that the optimality in the first hyper-period doesn't necessarily lead to the optimal solution in the steady state. But Lemma 3 shows the situation where our MFT algorithm presents the optimal solution in the steady state.

Lemma 3. Let S^*_{n',H,m^*} be the TFSS speed schedule with minimum number of missed jobs, m* (optimal solution in the steady state). Define $\Omega_{S^*_{n',H,m^*}}$ as the initial temperature of a hyper period in the steady state under schedule S^*_{n',H,m^*}. If MFT algorithm is run with initial temperature $\Omega_{S^*_{n',H,m^*}}$, then it certainly finds a LTF speed schedule with m* number of missed jobs which is also TFSS.

Proof: Note that in the steady state the temperature at the end and beginning of hyper-period is equal. Therefore for S^*_{n',H,m^*}, T_H (final temperature) is equal to $\Omega_{S^*_{n',H,m^*}}$.

Due to optimality of MFT algorithm for the first hyper-period, if we run MFT with $T_{init} = \Omega_{S^*_{n',H,m^*}}$, it will certainly

find a LTF speed schedule with m^* missed jobs and minimum final temperature, named Ω^*. Since we know that there exists a TFSS speed schedule (S^*_{n',H,m^*}) with m^* missed jobs and final temperature $\Omega_{S^*_{n',H,m^*}}$, therefore $\Omega^* \leq \Omega_{S^*_{n',H,m^*}}$. Thus we have a LTF speed schedule with m^* missed jobs, where $T_{init} = \Omega_{S^*_{n',H,m^*}}$ and $T_H = \Omega^* \leq \Omega_{S^*_{n',H,m^*}}$, which satisfies the conditions of Lemma 1. Then, it is also TFSS speed schedule.

To verify Lemma 3, for a specified job set, we have obtained the optimal speed schedule, S^*, in the steady state through an exhaustive search on all possible speed schedules. Setting T_{init} with Ω_{S^*} (hyper period initial temperature of S^* in steady state) in our MFT algorithm, the optimal result was obtained, which is as expected on the basis of lemma 3.

5. Quasi-Static Scheduling for Variable Ambient Temperature

In this section a quasi-static approach is used to manage variable ambient temperature. Thermal behavior of a speed schedule is dependent on the ambient temperature; therefore the ambient temperature variation might make a speed schedule invalid due to temperature constraint violation. To deal with such situation, we need to know maximum ambient temperature in which a speed schedule is valid. In fact, for a given range of ambient temperature $[T^{min}_{amb}, T^{max}_{amb}]$, a list of TFSS speed schedules is derived based on TFSS_SS, each of which are valid for the specified sub-interval of ambient temperature. The above mentioned declared list of speed schedules is obtained offline using a pseudo-polynomial algorithm, but an online algorithm is also required to properly handle transition between speed schedules. To this aim, we present a heuristic algorithm with low complexity which is based on DP approach that is presented in Section 6. Both offline and online phases of our approach are covered during this.

5.1. Offline Speed Scheduling

We define ambient temperature threshold, denoted by $ATT(S)$, for a speed schedule S, as the maximum ambient temperature in which S is valid, i.e. the temperature threshold T_{max} is not violated in steady state under S for all ambient temperatures bellow $ATT(S)$. To this aim, first we propose a method for calculating $ATT(S)$ for a given speed schedule. Then, we use this method beside TFSS_SS to generate a set of speed schedules such that for each ambient temperature, there is one TFSS speed schedule. This set is saved offline and is used at run time.

To calculate $ATT(S)$, we first need to compute the peak temperature of S in steady state as a function of ambient temperature. Suppose that the system is in steady state, therefore based on (2) we can derive the temperature at the end of job J_j in the steady state as:

$$T_j = \left(\sum_{i=1}^{j}\left(T_{s_i}\left(1-e^{-bt_i}\right)\right)e^{-b\left\lceil\frac{j-i}{j}\right\rceil\left(\sum_{k=i+1}^{j}t_k\right)}\right) + T_0 e^{-bt} \quad (7)$$

where

$$T_{s_i} = P_i R + T_{amb,} \quad (8)$$

and t_k is the execution time according to the speed determined by speed schedule S for job k, and $t = \sum_{i=1}^{j} t_i$. Note that in (8), P_i is the amount of power consumed by job J_i when it is executed under selected speed schedule. On the other hand, in the steady state the temperature at the beginning and end of hyper-period is the same so we have $T_{n'} = T_0$, which with replacement in (7), the initial temperature of hyper-period in steady state is obtained by:

$$T_0 = \frac{\sum_{i=1}^{n'}\left(T_{s_i}\left(1-e^{-bt_i}\right)\right)e^{-b\left\lceil\frac{n'-i}{n'}\right\rceil\left(\sum_{k=i+1}^{n'}t_k\right)}}{\left(1-e^{-b\sum_{i=1}^{n'}t_i}\right)} \quad (9)$$

Replacing (8) in (9), T_0 in steady state can be written as a linear function of T_{amb}:

$$T_0 = \frac{\sum_{i=1}^{n'}\left((P_i R + T_{amb})\left(1-e^{-bt_i}\right)\right)e^{-b\left\lceil\frac{n'-i}{n'}\right\rceil\left(\sum_{k=i+1}^{n'}t_k\right)}}{\left(1-e^{-b\sum_{i=1}^{n'}t_i}\right)} = \alpha + \beta\, T_{amb} \quad (10)$$

Applying relations (8) and (10) to (7), we can write the temperature at the end of job J_j as:

$$T_j = \alpha_j + \beta_j T_{amb} \quad (11)$$

For an ambient temperature T_{amb}, we define $T_{peak} = \max(T_j)$ as peak temperature of speed schedule S in steady state. Assume that for one j such as j^*, we have $T_{peak} = T_{j^*}$. By replacing T_{j^*} with T_{Max} in (11) which means the peak temperature be equal to maximum threshold temperature, and solving the equation for T_{amb}, we can finally calculate $ATT(S)$ as $ATT(S) = (T_{peak} - \alpha_j)/\beta_j$.

Input: $[T^{min}_{amb}, T^{max}_{amb}]$: range of ambient temperature variations, $J = \{J_1, J_2, \dots, J_n\}$: a job set, T_{max}: temperature threshold, T_{init}: initial temperature
Output: $SchSet$/* a set of TFSS speed schedules for each specified sub-interval of ambient temperature */
1. $T^p_{amb} = T^{min}_{amb}$
2. $S = TFSS\_SS(J, T_{init}, T^p_{amb}, T_{max})$
3. $T^m_{amb} = ATT(S)$;
4. $SchSet = \{\langle T^p_{amb}, T^m_{amb}, S\rangle\}$
5. **while** $(T^m_{amb} < T^{max}_{amb})$
6. $\quad T^p_{amb} = T^m_{amb}$
7. $\quad S = TFSS\_SS(J, T_{init}, T^p_{amb}, T_{max})$
8. $\quad T^m_{amb} = ATT(S)$
9. $\quad SchSet = SchSet \cup \{\langle T^p_{amb}, T^m_{amb}, S\rangle\}$
10. **return** $SchSet$

Figure 3. High level description of offline algorithm based on TFSS_SS to create a set of TFSS speed schedules for different ambient temperatures

Our proposed algorithm to create the set of speed schedules for different ambient temperatures is summarized in figure 3. In this algorithm, T^{min}_{amb} and T^{max}_{amb} are minimum and maximum possible ambient temperatures. We start by T^{min}_{amb}, then a TFSS speed schedule is derived according to

TFSS_SS for this ambient temperature (Line 2). After that, the maximum ambient temperature T^{m}_{amb} in which the selected speed schedule is valid is calculated (Line 3). Now the algorithm is repeated for T^{m}_{amb} to find another TFSS speed schedule. This process will be repeated until $T^{m}_{amb} \geq T^{max}_{amb}$.

5.2. Online Speed Schedule Switching

At runtime, whenever T_{amb} exceeds the ATT(S) of the current schedule, a speed schedule switching event is triggered. According to the offline table created by the above algorithm, the appropriate speed schedule is selected to be replaced with the current one. Note that in this study it is assumed that the ambient temperature variation occurs at the steady state, because the variation of ambient temperature occurs in a large interval than the system temperature.

It should be considered that if the ambient temperature variation occurred within hyper-period (i.e. not exactly at the end of hyper-period) such that a speed schedule switching is required, then continuing with the current schedule until the end of hyper-period may cause the temperature at the end of current hyper period reaches a value that is not acceptable for the new speed schedule, resulting in violation of T_{max} for some hyper-periods until the system reaches steady state. Therefore for each speed schedule S we need to know the maximum feasible initial temperature, MFIT(S), in which S could be applied without temperature constraint violation. A maximum feasible initial temperature is defined as:

MFIT(S) = max {T | running S with initial temperature T does not violate T_{max}}.

According to (7), it is straightforward to obtain MFIT(S) of a given speed schedule S.

We propose an online heuristic algorithm in order to bring the processor temperature below MFIT(S) such that newly selected speed schedule can start to work without violating temperature constraint. To this aim, we solve the following problem:

Problem 2. Given a set of FRT jobs J in a specified time interval which are scheduled by a non-preemptive scheduling policy, an ambient temperature T_{amb}, initial temperature T_{init}, a maximum allowable temperature T_{max}, and a desired final temperature T_f find a thermally feasible speed schedule S for that interval which has minimum missed jobs and satisfies the following condition:

$$T_{final}(S) \leq T_f \tag{12}$$

where $T_{final}(S)$ is the temperature at the end of time interval with speed schedule S .

Let $S_{i,m}$ be the speed schedule for the first i jobs that produces minimum final temperature where exactly m jobs of i jobs are missed during a hyper-period. We denote the final temperature produced by $S_{i,m}$ as $T(i, m)$.

If there is not any feasible solution for $T(i, m)$, then we define it as ∞. We propose the following recursive formula to calculate $T(i, m)$:

$$T_{exe} = \{T(i-1, m) + \Delta T(s_k) |$$
$$r_i \leq ct_{i-1}, ct_{i-1} + c_{ik} \leq D_i, T(i-1, m) \neq \infty, k \in [1, K]\}$$
$$T_{miss} = \{T(i-1, m-1) \mid r_i \leq ct_{i-1}\}$$
$$T(i, m) = \text{Min}\{t | t \in T_{exe} \cup T_{miss}, t < T_{max}\} \tag{13}$$

where ct_i is the completion time of the ith job. It is important to note that this formula does not produce an optimal result. However, it is a heuristic approach to calculate a result that satisfies the desired requirements with a rather low time and space complexity.

Every cell $T(i, m)$ has an entry of the minimum final temperature when i jobs are executed and m jobs of i jobs are missed. For each cell in the table we save the completion time of the last job and the speed mode that leads to the respective temperature. So, in addition to temperature table a frequency table is also constructed with the same dimension.

The table is filled as follow: First all the cells on the diameter are set as T_{init}, i.e.:

Input: SchSet: set of TFSS speed schedules for different interval of ambient temperatures; T_{cur}: current temperature of the system; T_{amb}: new ambient temperature; $[T^{min}_{amb}, T^{max}_{amb}]$:range of ambient temperature variations, J: a job set, T_{max}:temperature threshold,

1. find$\langle T^{p}_{amb}, T^{m}_{amb}, S\rangle$ from SchSet such that $T^{p}_{amb} \leq T_{amb} \leq T^{m}_{amb}$.
2. $T_f = MFIT(S)$
3. S'=SS_Sw(J,T_{amb} ,T_{cur},T_f)
4. Employ S' until the end of current hyper-period, then employ S.

Figure 4. High level description of online speed schedule switching method

$$T(i, i) = T_{init} \tag{14}$$

The first column of the table, $T(i, 0)$ that denotes i jobs are executed without any miss is filled according to formula (13), then the other cells will be filled row by row.

At last, the frequency table presents several speed schedules; one for each number of jobs misses. We choose the speed schedule as a final solution that has a minimum number of miss and the final temperature at the end of hyper period be less than or equal to T_f. That is, we find S_{n',m^*} where

$$m^* = \arg \text{Min}_m \{m | T(n', m) \leq T_f\} \tag{15}$$

Then, the system is run under the found speed schedule for a specified interval, and then the offline computed speed schedule that is TFSS can be safely utilized. We refer to this algorithm as SS_Sw which gets a job set, the current system and ambient temperature and final temperature that should be reach at the end of current hyper-period, as input. Figure 4 summarizes our online speed schedule switching method. Steps presented in figure 4 are used when the speed schedule switching event is triggered in the middle of a hyper-period. Otherwise, there is no need to use SS_Sw and the current speed schedule is simply replaced with the appropriate speed schedule selected from the offline computed table (derived by method described in figure 3).

6. Evaluation

In this section, the effectiveness of the proposed algorithm MFT, and quasi-static scheduling will be evaluated using a number of experiments. We compare our algorithm with the following scheduling algorithms:

- Non-preemptive EDF (NP-EDF) that runs the tasks with maximum speed and brings the CPU to sleep mode if there is no workload. When executing one task violates the temperature threshold, it is not executed at all.
- Reactive two-speed scheduling [12, 13], in which the processor runs with the maximum processor speed while there are jobs to execute and the temperature is below the threshold. If the temperature reaches to the threshold, the processor works at the equilibrium speed such that the temperature never exceeds the threshold. Note that the equilibrium speed is not necessarily one of the available speed modes. Therefore we use the nearest available speed mode of the processor.

In our experiments, we consider a FRT system, where if one job violates either its temperature or timing constraint, will not be executed at all and thrown away from the system.

It is considered that the processor has four speed modes, {0.5, 0.66, 0.83, 1} besides one sleep mode which consumes no energy. The thermal capacitance is chosen as 140.3 J/°C based on the Hotspot simulator and the thermal resistance is set to 0.7 ° C/W [17]. Also, the maximum temperature constraint and the ambient temperature are set as 100°C and 45°C, respectively.

Each experiment is repeated 30 times, each of them is called a run, and the target measures are evaluated based on their average values over these runs. It is worth to mention that MFT algorithm needs the non-preemptive schedule of the jobs in a job set. So the first step of each experiment is to generate such schedule, regardless of temperature constraints. We have employed non-preemptive EDF in this phase.

There are three sets of experiments with various system utilization, various initial temperatures and variable ambient temperature. The target measure is the ratio of missed jobs to the total jobs in the steady state.

6.1. Scenario 1: Various System Utilizations

In the first scenario, our proposed algorithms are evaluated according to the miss ratio in the steady state. We assume a fixed ambient temperature similar to [11] and perform our experiment for different system utilization. The initial temperature is set to ambient temperature. System utilization u is varied from 0.2 to 1 (with steps of 0.2). For each utilization u, 30 task sets are generated. Task periods and utilizations are randomly generated according to uniform distribution. After that the execution time of the task can be calculated.

Figure 5 shows miss ratio of three speed scheduling algorithms in the steady state. As it has been illustrated, MFT works better than the others. Trend of the curves agrees with the fact that in lower utilizations, the processor will remain idle most of the times which decreases the temperature. Thus, reactive two-speed algorithm is the same as MFT in utilizations below 0.6 and in higher utilizations we will see rise in the curves.

Since the NP-EDF has no temperature consideration, it is not surprising that the miss ratio of this algorithm is significantly higher than the others.

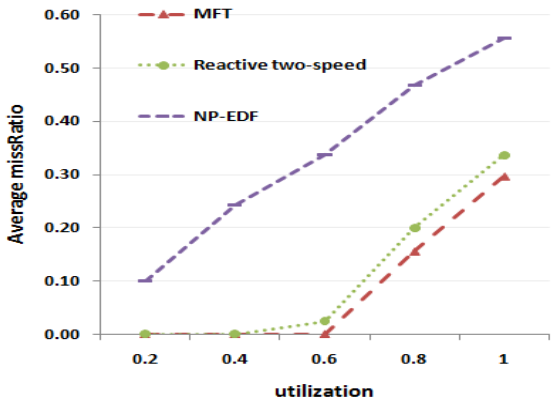

Figure 5. Miss ratio of the algorithms in the steady state

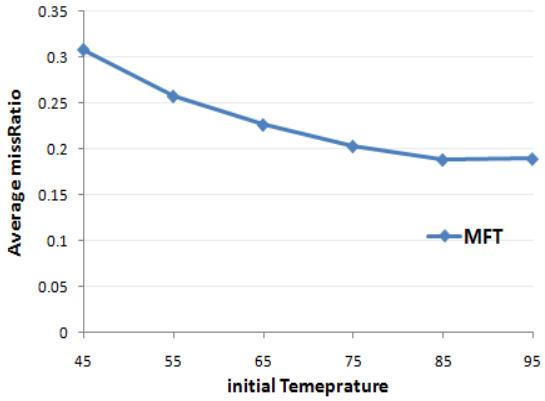

Figure 6. Effect of initial temperature on miss ratio

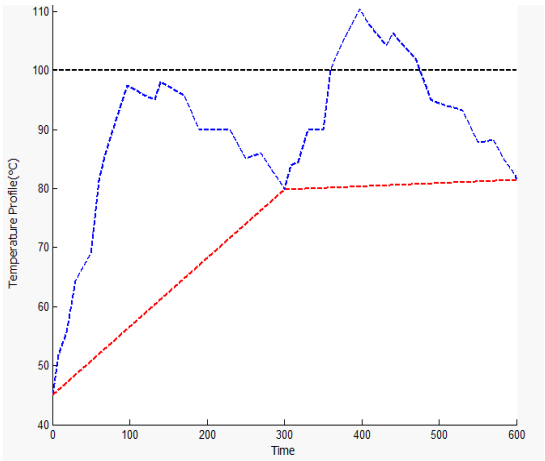

Figure 7. Temperature profile of the MFT in the first two hyper-periods when $T_{init} = 45$ (red line shows the initial temperature at the start of consecutive hyper-periods)

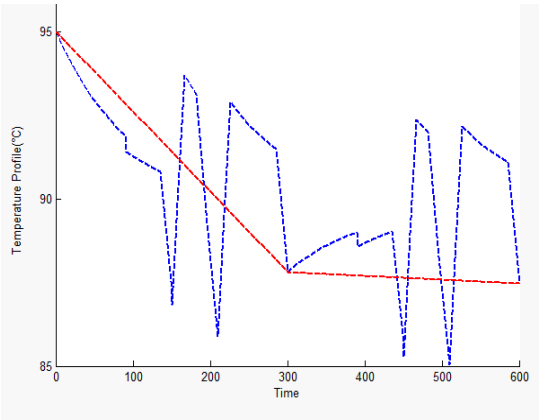

Figure 8. Temperature profile of the MFT in the first two hyper-periods when $T_{init} = 95$ (red line shows the initial temperature at the start of consecutive hyperperiods)

6.2. Scenario 2: Various Initial Temperature

In the second scenario we evaluate the effect of initial temperature on the miss ratio of jobs. We consider a set of task sets, with utilization 0.8. The initial temperature is varied from 45°C to 95°C with step 10°C. As it is shown in figure 6, increasing the initial temperature generally leads to decrease in miss ratio of speed schedules and therefore, get a better performance.

To justify results in figure 6, we consider one job set from the aforementioned job sets which is scheduled based on MFT algorithm with two different initial temperatures. Figure 7 illustrates the temperature behavior in the two first hyper-periods when $T_{init} = 45$ and figure 8 shows the same result when $T_{init} = 95$. Based on these figures, for a specified number of misses, the speed schedule that is achieved with $T_{init} = 95°C$ is TFSS, because it is more conservative since the temperature is near to the temperature threshold. In contrast, for the same number of misses, the speed schedule which is obtained by $T_{init} = 45°C$ violates the temperature constraint in the second hyper-period. In fact a low initial temperature leads to high value of peak temperature in the first hyper-period. In contrast, with the higher initial temperature, the scheduling algorithm selects speed levels leading to smooth temperature variation in the first hyper-period which can help in achieving a TFSS speed schedule.

It seems that the initial temperature $T_0 = T_{max}$ generates the schedule with minimum number of missed jobs, but it may not always be true. Based on the job set, for high value of T_0 (close to T_{max}), the processor may need to be idle or execute in a low speed at the beginning of hyper-period in order to prevent thermal violation. This fact may cause some jobs miss their deadline, which could be avoided if T_0 had a lower value.

6.3. Scenario 3: Variable Ambient Temperature

In this scenario the quasi-static scheduling algorithm is evaluated which is designed for the variable ambient temperature. We consider the ambient temperature variation occurs at the steady state and assume that the ambient temperature varies with an incremental pattern. We set T_{amb}^{min} and T_{amb}^{max} as 40°C and 60°C, respectively. In this experiment the rate of ambient temperature changes is almost every 1200 time units. Figure 9 shows the thermal profile of one job set with utilization 1, and initial temperature 40 ° C, under ambient temperature variation.

Note that, increasing ambient temperature leads to increasing steady state temperature and therefore, changes the thermal behavior of current speed schedule. In figure 9 we start with the speed schedule that is determined for T_{amb}^{min}. As it is seen, the speed schedule is valid for several ambient temperatures until time 5000 where the respective schedule is not valid for such ambient temperature because if the system continues this speed schedule, the temperature constraint is certainly violated. Therefore the algorithm selects another speed schedule which is valid for the new situation. This process will be repeated while the ambient temperature varies.

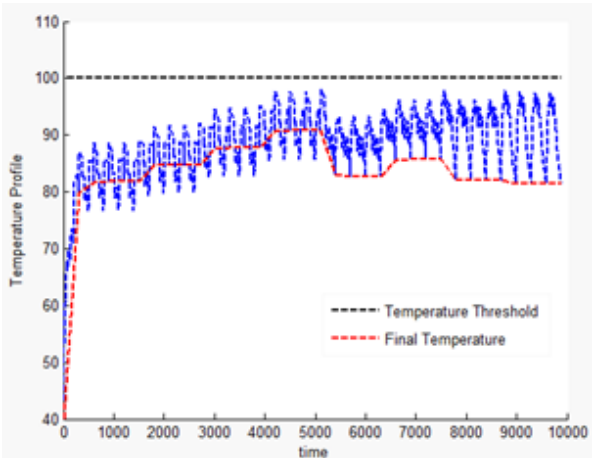

Figure 9. Temperature profile of system under ambient temperature variation

7. Conclusion

In this paper, we have presented a proactive speed scheduling for periodic FRT systems under the maximum temperature constraint to minimize the jobs missing their deadlines. To obtain a thermally feasible speed schedule for the periodic system, two steps are accomplished. For the first step, we proposed a temperature-aware algorithm, MFT which are based on the dynamic programming approach, to achieve speed schedules that are thermally feasible in the first hyper-period.

At the second step, the speed schedule that has a minimum number of missed jobs and is thermally feasible in the steady state is selected as a final solution. We also present a quasi-static approach for variable ambient temperature where a number of speed schedules, each for a range of ambient temperatures, are generated offline. At runtime with the temperature variation, the appropriate speed schedule is selected to be used by the system.

The experimental results demonstrate that our proposed algorithm effectively reduce the miss ratio measure in high utilizations comparing to reactive two-speed and NP-EDF scheduling algorithms.

References

[1] M. Mohaqeqi, M. Kargahi, and A. Movaghar, "Analytical Leakage-aware Thermal Modeling of a Real-Time System," *IEEE Trans. on Computers*, vol. 63, no. 6, pp. 1378-1392, 2014.

[2] R. Jayaseelan, and T. Mitra, "Temperature-aware Scheduling for Embedded Processors," *Proc. IEEE Intl Conf. on Very Large Scale Integration Design*, pp. 541-546, 2009.

[3] X. Fu, X. Wang, and E. Puster, "Dynamic Thermal and Timeliness Guarantees for Distributed Real-Time Embedded Systems," *Proc. IEEE Intl Conf. on Embedded and Real-Time Computing Systems and Applications*, pp. 403-412, 2009.

[4] L. Yuan, and G. Qu, "Alt-DVS: Dynamic Voltage Scaling with Awareness of Leakage and Temperature for Real-Time Systems," *Proc. IEEE Intl Conf. on Adaptive Hardware and Systems*, pp. 660-670, 2007.

[5] S. Wang, and R. Bettati, "Delay Analysis in Temperature Constrained Hard Real-Time Systems with General Task Arrivals," *Proc. IEEE Intl Symp. on Real-Time Systems*, pp. 323-334, 2006.

[6] K. Skadron, M. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan, *Temperature-aware Micro-architecture: Extended Discussion and Results*, Technical Report, University of Virginia, Charlottesville, Virginia, 2003.

[7] D. Brooks, and M. Martonosi, "Dynamic Thermal Management for High-Performance Microprocessors," *Proc. IEEE Intl Symp. on High-Performance Computer Architecture*, pp. 171-182, 2001.

[8] M. Mohaqeqi, M. Kargahi, and A. Movaghar, "Analytical Leakage/Temperature-aware Power Modeling and Optimization for a Variable Speed Real-Time System," *Proc. IEEE Intl Conf. on Real-Time and Network Systems*, pp. 81-90, 2012.

[9] A. K. Coskun, T. Rosing, and K. Whisnant, "Temperature aware Task Scheduling in MPSoCs," *Proc. IEEE Intl Conf. on Design*, *Automation and Test in Europe*, pp. 1-6, 2007.

[10] N. Bansal, T. Kimbrel, and K. Pruhs, "Speed Scaling to Manage Energy and Temperature," *ACM Transaction*, vol. 54, no. 1, pp. 1-39, 2007.

[11] S. Wang, and R. Bettati, "Reactive Speed Control in Temperature Constrained Real-Time Systems," *Journal of Real-Time Systems*, vol. 39, no. 1, pp. 658-671, 2008.

[12] J. J. Chen, S. Wang, and L. Thiele, "Proactive Speed Scheduling for Real-Time Tasks under Thermal Constraints," *Proc. IEEE Intl Symp. Real-Time and Embedded Technology and Applications*, pp. 141-150, 2009.

[13] D. Bertozzi, A. Acquaviva, D. Bertozzi, and A. Poggiali, "Supporting Task Migration in Multi-Processor Systems-on-Chip: A Feasibility Study," *Proc. IEEE Intl Conf. on Design*, *Automation and Test in Europe*, pp. 1-6, 2006.

[14] P. Kumar, and L. Thiele, "End-to-End Delay Minimization in Thermally Constrained Distributed Systems," *Proc. IEEE Intl Conf. on Real-Time Systems*, pp. 81-91, 2011.

[15] P. Kumar, and L. Thiele, "Cool Shapers: Shaping Real-Time Tasks for Improved Thermal Guarantees," *Proc. IEEE Intl Conf. on Design*, *Automation and Test in Europe*, pp. 468-473, 2011.

[16] M. Hettiarachchi, N. Fisher, M. Ahmed, L. Wang, Sh. Wang, and W. Shi, "The Design and Analysis of Thermal-Resilient Hard Real-Time Systems," *Proc. IEEE Intl Symp. Real-Time and Embedded Technology and Applications*, pp. 67-76, 2012.

[17] S. Zhang, and K. S. Chatha, "Approximation Algorithm for the Temperature-aware Scheduling Problem," *Proc. IEEE Intl Conf. on Computer-Aided Design*, pp. 281-288, 2007.

[18] S. Wang, and R. Bettati, "Reactive Speed Control in Temperature-Constrained Real-Time Systems," *Proc. IEEE Intl Conf. on Real-Time Systems*, pp. 161-170, 2006.

[19] S. Wang, J. J. Chen, Z. Shi, and L. Thiele, "Energy-Efficient Speed Scheduling for Real-Time Tasks under Thermal Constraints," *Proc. IEEE Intl Conf. on Embedded and Real-Time Computing Systems and Applications*, pp. 201-209, 2009.

[20] V. Chaturvedi, H. Huang, and G. Quan, "Leakage-aware Scheduling on Maximal Temperature Minimization for Periodic Hard Real-Time Systems," *Proc. IEEE Intl Conf. on Computer and Information Technology*, pp. 1802-1809, 2010.

[21] G. Quan, and V. Chaturvedi, "Feasibility Analysis for Temperature Constraint Hard Real-Time Periodic Task," *IEEE Trans. on Industrial Informatics*, vol. 6, no. 3, pp. 329-339, 2010.

[22] G. Quan, Y. Zhang, W. Wiles, and P. Pei, "Guaranteed Scheduling for Repetitive Hard Real-Time Tasks under the Maximal Temperature Constraint," *Proc. IEEE Intl Conf. on*

Hardware/Software Co-design and System Synthesis, pp. 267-272, 2008.

Morteza Mohaqeqi received his B.S., M.S., and PhD degrees in computer engineering from the University of Tehran in 2008, 2010, and 2015, respectively. His research interests include power/thermal-aware scheduling and resource management in distributed and real-time systems.
E-mail: m.mohaqeqi@alumni.ut.ac.ir

Mehdi Kargahi (M'09, SM'13) received the BS degree in computer engineering from Amir-Kabir University of Technology (Tehran Poly-Techniques) in 1998 and the MS and PhD degrees in computer engineering from Sharif University of Technology in Tehran in 2001 and 2006, respectively. He is currently an associate professor in the Department of Electrical and Computer Engineering at the University of Tehran, Iran. He has been a researcher in the School of Computer Science at the Institute for Studies in Theoretical Physics and Mathematics (IPM) from 2003. His research interests include performance modeling and power management of dependable real-time systems with stochastic properties. He is a senior member of the IEEE.
E-mail: kargahi@ut.ac.ir, kargahi@ipm.ir

Fatemeh Gharehdaghi received her B.S. and M.S. degrees in computer engineering from the University of Tehran in 2010 and 2013, respectively. Her research interests is power/thermal-aware scheduling in real-time systems.
E-mail: f.gharedaghi@ut.ac.ir

Paper Handling Data:

Submitted: 04.05.2015
Received in revised form: 10.06.2015
Accepted: 20.06.2015
Corresponding author: Dr. Mehdi Kargahi,
School of Electrical and Computer Engineering, University of Tehran, Tehran, Iran, Iran.