Dynamic Thermal and Timeliness Guarantees for Distributed Real-Time Embedded Systems*

Xing Fu, Xiaorui Wang, and Eric Puster
Department of Electrical Engineering and Computer Science
University of Tennessee, Knoxville, TN 37996, USA
{*xfu1, xwang, epuster*}*@utk.edu*

Abstract

Distributed real-time embedded systems have stringent requirements for key performance properties, such as end-to-end timeliness and reliability, in order to operate properly. In recent years, with the continuously decreasing feature size and increasing demand for computation capabilities, today's real-time embedded systems face an increasing probability of overheating and even thermal failures. As a result, their temperature must be explicitly controlled for improved reliability. While a variety of control algorithms have been proposed for either real-time guarantees or thermal management in an isolated *manner, this paper proposes a coordinated control solution that can provide simultaneous thermal and timeliness guarantees for distributed real-time embedded systems running in unpredictable environments. Our control solution features a novel coordination design, which allows the thermal and timeliness control loops to run on their respective desired small timescales for prompt control actions and yet achieve theoretically guaranteed control accuracy and system stability. In addition, while most existing work relies solely on simulations, we present empirical results on a physical testbed to demonstrate the efficacy of our control solution.*

1 Introduction

In recent years, a new class of real-time applications called distributed real-time embedded (DRE) systems has been rapidly growing. Different from traditional real-time systems that run in closed execution environments, DRE systems commonly execute in open and *unpredictable* environments, in which both workloads and system conditions are unknown and may vary significantly at runtime. For example, task execution times in vision-based surveillance systems depend on the content of live camera images of changing environments [1]. Traditional approaches to handling real-time tasks rely on schedulability analysis in an open-loop manner, and thus cannot be directly applied to DRE systems, because they may violate the desired timing constraints or severely under-utilize the system when task execution times are highly unpredictable. Therefore, DRE applications commonly require runtime control and guarantees of end-to-end timeliness for their proper operation.

Feedback control techniques have recently shown a lot of promise in providing runtime real-time guarantees for DRE systems by adapting to workload variations based on dynamic feedback. In particular, feedback-based *CPU utilization control* [2][3] has been demonstrated to be an effective way of meeting the *end-to-end deadlines* for soft DRE systems. The (CPU) utilization of a processor is the percentage of time when its CPU performs useful computation, and is an important performance metric for many computer systems. The goal of utilization control is to enforce appropriate schedulable utilization bounds (*e.g.*, Liu and Layland bound) on all the processors in a DRE system, despite significant uncertainties in system workloads. As a result, utilization control can meet all the end-to-end real-time deadlines of the DRE system without accurate knowledge of the workload such as task execution times, as well as when execution times vary within limited ranges at runtime. In the meantime, utilization control can maximize the system utility by controlling CPU utilizations to stay slightly below their schedulable bounds so that the processors can be utilized to the maximum degree. Utilization control can also enhance system reliability by providing overload protection against workload fluctuation [4].

However, existing work on utilization control can only provide timeliness guarantees, while today's DRE systems face an increasing probability of overheating and even thermal failures, due to their continuously decreasing feature size and increasing demand for computation capabilities. For example, recent studies show that 50% of all electronics failures are related to overheating [5]. More specifically, the lifetime of a processor can be approximately halved if it runs 10-15°C higher than its normal temperature range [6]. Furthermore, a 15°C increase in temperature could double the failure rate of a disk drive [7]. Therefore, thermal constraints also need to be strictly enforced for DRE systems. Although some recent research has proposed optimization algorithms based on task allocation and configurations of processor voltage/frequency to achieve minimized system temperature and guaranteed real-time performance [8, 9], those open-loop solutions cannot be directly applied to DRE sys-

*This work was supported, in part, by NSF under grants CNS-0720663 and CNS-0845390, and by ONR under grant N00014-09-1-0750.

tems where workloads and system conditions may vary at runtime. While some dynamic thermal managment (DTM) approaches have been proposed for general computer systems (*e.g.*, [10]), they cannot provide desired real-time guarantees for DRE systems. Therefore, existing work can only provide either timeliness guarantees or thermal control in an *isolated* manner.

Simultaneous thermal and utilization control is challenging because the desired guarantees *cannot* be achieved by simply putting the two control loops together. Without effective coordination, individual control solutions may conflict with each other. For example, many thermal management methods rely on dynamic voltage and frequency scaling (DVFS), which may significantly impact the execution times of the real-time tasks running in the systems. As a result, the timeliness guarantees provided by existing control solutions may be severely violated. In addition, although each control loop can be proven to be stable individually, system stability must be theoretically guaranteed for the entire system. Although previous work has approached the coordination problem by forcing one control loop to run on a significantly longer timescale than the other loop [11], both the thermal and utilization control loops must run on small timescales for DRE systems, because both the thermal and timing constraints are critical and must be promptly enforced upon any violations. Hence, a new kind of coordination methodology must be designed and analyzed.

This paper proposes a novel coordinated thermal and utilization control solution, based on robust control theory [12], to provide simultaneous thermal and timeliness guarantees for DRE systems. The thermal control loop locally controls the temperature of each processor, while the utilization control loop provides end-to-end timeliness guarantees at the cluster level. Specifically, the main contributions of this paper are three-fold:

- While most existing work relies on open-loop optimization to minimize power/temperature for DRE systems with the assumption that task execution times and system thermal condition do not change significantly at runtime, we analytically model the temperature and CPU utilizations of a DRE system and design a feedback control solution for dynamic thermal and real-time guarantees for DRE systems running in *unpredictable* environments.

- While most existing closed-loop solutions provide either thermal or timeliness guarantee in an isolated manner, our solution coordinates the thermal and utilization control loops to provide simultaneous runtime guarantees. To our best knowledge, our solution is the first one that adopts robust control theory as a theoretical foundation such that both control loops can run on their respective desired timescales for prompt control actions with guaranteed system stability.

- While most existing work relies solely on simulations for evaluation, we present empirical results on a physical testbed to demonstrate the efficacy of our control solution.

The rest of this paper is organized as follows. Section 2 presents the control architecture. Section 3 briefly introduces the utilization control loop while Section 4 provides the detailed design and analysis of the thermal control loop. Section 5 discusses the coordination of the two control loops. Section 6 introduces the implementation of the control solution. Section 7 presents our empirical results on our physical testbed. Section 8 reviews the related work. Finally, Section 9 summarizes the paper.

2 Coordinated Control Solution

In this section, we introduce our task model and the coordinated control architecture.

2.1 Task Model

We adopt an end-to-end task model [13] implemented by many DRE applications. A system is comprised of m periodic tasks $\{T_i|1 \leq i \leq m\}$ executing on n processors $\{P_i|1 \leq i \leq n\}$. Task T_i is composed of a set of sub-tasks $\{T_{ij}|1 \leq j \leq n_i\}$ which may be located on different processors. A processor may host one or more sub-tasks of a task. The release of subtasks is subject to precedence constraints, i.e., subtask $T_{ij}(1 < j \leq n_i)$ cannot be released for execution until its predecessor subtask T_{ij-1} is completed. All the subtasks of a task share the same rate. The rate of a task (and all its subtasks) can be adjusted by changing the rate of its first subtask. If a non-greedy synchronization protocol (e.g., release guard [14]) is used to enforce the precedence constraints, every subtask are released periodically without jitter.

In our task model, each task T_i has a *soft* end-to-end deadline related to its period. In an end-to-end scheduling approach [14], the deadline of an end-to-end task is divided into subdeadlines of its subtasks. Hence the problem of meeting the end-to-end deadline can be transformed to the problem of meeting the subdeadline of each subtask. A well known approach for meeting the subdeadlines on a processor is to ensure its utilization remains below its schedulable utilization bound [13].

Our task model has two important properties. First, while each subtask T_{ij} has an *estimated* execution time c_{ij} available at design time, its *actual* execution time may be different from its estimation and vary at run time. Modeling such uncertainty is important to DRE systems operating in unpredictable environments. Second, the rate of a task T_i may be dynamically adjusted within a range $[R_{min,i}, R_{max,i}]$. This assumption is based on the fact that the task rates in many applications (e.g., digital control, sensor update, and multimedia) can be dynamically adjusted without causing system failure. The rate ranges are determined by the applications (e.g., the limited sampling frequency of a sensor). A task

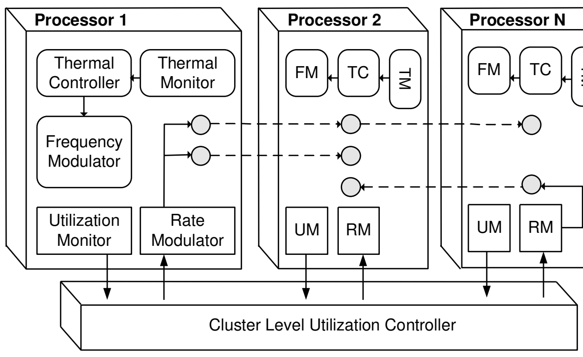


Figure 1. Coordinated Control Architecture

running at a higher rate contributes a higher value to the application at the cost of higher utilizations.

2.2 Control Architecture

As shown in Figure 1, our coordinated solution includes a cluster-level utilization control loop and a thermal control loop on each processor.

The cluster-level utilization control loop dynamically controls the utilizations of all the processors by adjusting task rates within their allowed ranges. Because the rate change of a task affects the utilizations of all the processors where the task has subtasks, this loop is a Multi-Input-Multi-Output (MIMO) control loop, which works as follows: (1) the utilization monitor on each processor P_i sends its utilization $u_i(k)$ in the last control period to the cluster-level controller; (2) the controller computes a new rate $r_j(k)$ for every task T_j and sends the new rates to the rate modulators; and (3) the rate modulators change the task rates accordingly.

On every processor P_i in the system, we have a local controller that controls the processor temperature by scaling the DVFS level of the processor. The controller is a Single-Input-Single-Output (SISO) controller because we assume the CPU frequency change of P_i only affects the temperature of P_i. This is usually true because different processors in a DRE system may locate in different places. This loop works as follows: (1) the thermal monitor on P_i sends its temperature $t_i(k)$ to the local thermal controller; (2) the controller computes a new DVFS level $f_i(k)$ and sends it to the frequency modulator on P_i; and (3) the frequency modulator changes the processor DVFS accordingly.

It is clear that without effective coordination, the two control loops may conflict with each other. For example, the thermal control loop relies on DVFS to control processor temperature. DVFS can significantly impact the execution times of the real-time tasks running in the systems and even cause the execution times to vary outside their stability ranges. As a result, the timeliness guarantees provided by the utilization control loop can be severely violated. On the other side, the CPU utilization changes made by the cluster-level utilization control loop may also impact the temperatures of multiple processors. Therefore, the coordination of the two control loops must be designed based on robust control theory to achieve global stability for the entire system. The detailed coordination is discussed in Section 5.

Since the core of each control loop is its controller, we introduce the design and analysis of the two controllers in the next two sections, respectively. The implementation details of other components are given in Section 6.

3 Utilization Control Loop

In this section, we briefly introduce the system model and design of the cluster-level utilization control loop.

3.1 System Modeling

We now establish a dynamic model that characterizes the relationship between the controlled variable $\mathbf{u}(k)$ and the manipulated variable $\mathbf{r}(k)$. We first model the utilization $u_i(k)$ of one processor P_i. The *estimated* utilization change $\Delta b_i(k)$ of P_i in the k^{th} control period can be modeled as a function of the execution times of all the subtasks on P_i and their rate changes $\Delta r_j(k) = r_j(k) - r_j(k-1)$.

$$\Delta b_i(k) = \sum_{T_{jl} \in S_i} c_{jl} \Delta r_j(k) \tag{1}$$

where S_i is the set of subtasks located at processor P_i.

$\Delta b_i(k)$ is based on the estimated execution time c_{jl}. Since the actual execution times may be different from their estimation due to workload variations, we model the actual utilization of P_i, $u_i(k)$, as the following difference equation.

$$u_i(k+1) = u_i(k) + g_i \Delta b_i(k) \tag{2}$$

where the utilization gain g_i represents the ratio between the change to the actual utilization and its estimation $\Delta b_i(k)$. For example, $g_i = 2$ means that the actual change to utilization is twice the estimated change. Note that the exact value of g_i is *unknown* at design time due to the unpredictability of subtasks' execution times.

Note that in (2), we assume that the relative CPU frequency of P_i is 1, which means that the processor is running at its highest CPU frequency. However, since the thermal controller on P_i may use DVFS to control the processor temperature, the relative CPU frequency can become smaller than 1 at runtime. The impact of the thermal controller on the utilization control loop is analyzed in Section 5. Based on (2), a DRE system with m tasks and n processors is described by the following MIMO dynamic model.

$$\mathbf{u}(k) = \mathbf{u}(k-1) + \mathbf{G}\mathbf{\Delta b}(k-1) \tag{3}$$

where $\mathbf{G}$ is a diagonal matrix where $g_{ii} = g_i$ $(1 \leq i \leq n)$ and $g_{ij} = 0$ $(i \neq j)$. $\mathbf{\Delta b}(k)$ is a vector including the estimated utilization change (1) of each processor. The relationship between the utilization and task rates is characterized as follows:

$$\mathbf{\Delta b}(k) = F \Delta r(k) \tag{4}$$

The subtask allocation matrix, F, is an $n \times m$-order matrix, where $f_{ij} = c_{jl}$ if subtask T_{jl} (the lth subtask of task T_j) is allocated to processor i, and $f_{ij} = 0$ if no subtask of task T_j is allocated to processor i.

3.2 Controller Design

In this paper, we adopt the EUCON algorithm presented in our previous work [2] for utilization control. EUCON features a Model Predictive Controller (MPC) that optimizes a *cost function* defined over P control periods in the future, called the *prediction horizon*. The control objective is to select control inputs in the following M control periods, called *control horizon*, which minimizes the following cost function while satisfying the constraints.

$$V(k) = \sum_{i=1}^{P} \|\mathbf{u}(k+i|k) - \mathbf{ref}(k+i|k)\|^2 + \sum_{i=0}^{M-1} \|\mathbf{\Delta r}(k+i|k) - \mathbf{\Delta r}(k+i-1|k)\|^2 \quad (5)$$

The first term in the cost function represents the *tracking error*, i.e., the difference between the utilization vector $\mathbf{u}(k+i|k)$ and a reference trajectory $\mathbf{ref}(k+i|k)$ defined in [2]. By minimizing the tracking error, the closed-loop system will converge to the utilization set points if the system is stable. The second term in the cost function represents the control penalty. This control problem is subject to the task rate constraints . The detailed design and analysis of EUCON are available in [2].

Although the utilization control loop is proven to be stable in [2], in order for the coordinated solution to be stable, the stability and utilization control loop need to be reexamined with the impact from the thermal control loop. The coordination analysis is presented in Section 5.

4 Thermal Control Loop

In this section, we model, design, and analyze the thermal control loop.

4.1 System Model

We now model the temperature of a processor P_i. We first introduce some notation. T_s is the control period. $t_i(k)$ is the temperature of P_i in the k^{th} control period. $f_i(k)$ is the DVFS level of P_i in the k^{th} control period. $d_i(k)$ is the difference between $f_i(k)$ and $f_i(k-1)$, i.e., $d_i(k) = f_i(k) - f_i(k-1)$. $p_i(k)$ is the power consumption of P_i in the k^{th} control period. The control goal is to guarantee that $t_i(k)$ converges to the temperature set point in a finite settling time.

We use two steps to model the relationship between $t_i(k)$ and $f_i(k)$. In the first step, we analytically model the relationship between $t_i(k)$ and $p_i(k)$. In the second step, we model the relationship between $p_i(k)$ and $f_i(k)$.

First, since DVFS changes the frequency of the entire processor chip, we adopt a chip-level thermal model called resistor-capacitor model (RC-model) [15] to model the the processor temperature. To convert the thermal model in the continuous time domain to a model in the discrete time domain, the sampling rate (i.e., control period T_s) must be selected carefully to guarantee the precision of the discrete model. In this paper, we select the sampling rate less than the thermal time constant in the second-order circuits. Based on the model in [15], our thermal model is:

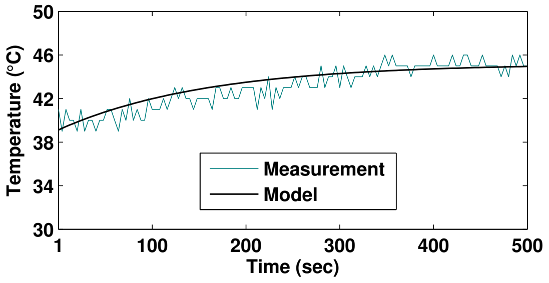


Figure 2. Model Prediction vs. Measurement

$$\Delta t_i(k) = \frac{p_i(k) \cdot T_s}{C_i} - \frac{t_i(k) \cdot T_s}{R_i \cdot C_i} \quad (6)$$

where $\Delta t_i(k) = t_i(k+1) - t_i(k)$. C_i and R_i are processor P_i's thermal capacitance and thermal resistance, respectively. Note that C_i and R_i are determined by the thermal characteristics of the CPU packaging of P_i and the cooling system. The temperature is related to the ambient temperature T_a. We can transform (6) to the following difference equation:

$$t_i(k+1) = (1 - \frac{T_s}{R_i C_i}) \cdot t_i(k) + \frac{p_i(k) \cdot T_s}{C_i} \quad (7)$$

In the second step, we model the relationship between processor power consumption $p_i(k)$ and $f_i(k)$. It is well-known that DVFS can allow cubic reductions in power density relative to performance loss in a processor [16]. However, a cubic power model may lead to high complexity for controller design and large runtime overhead. On the other hand, real processors usually only provide a limited DVFS range. Within the small range, previous studies [17, 18] have shown that the relationship between power and DVFS level can be approximated with a linear function.

$$p_i(k) = A_i f_i(k) + B_i \quad (8)$$

where A_i and B_i are generalized parameters that may vary for different processors. To determine the values of A_i and B_i, we can use a standard approach to this problem called system identification [19]. In this paper, we use a typical real-time system CPU computation intensive workload presented in [20] for system identification.

We now substitute (8) into (7) to establish relationship between $t_i(k)$ and $f_i(k)$. The dynamic model of the system as a difference equation is:

$$t_i(k) = (1+\Theta_i)t_i(k-1) - \Theta_i t_i(k-2) + \Psi_i d_i(k-1) \quad (9)$$

where $\Theta_i = (1 - \frac{T_s}{R_i \cdot C_i})$ and $\Psi_i = \frac{T_s}{C_i} \cdot A_i$.

We then use a step-like signal to validate our system model (9) on our physical testbed. Figure 2 demonstrates that the predicted output of the our model is sufficiently close to the measured actual system output.

4.2 Controller Design

Following standard control theory [19], we design a Proportional (P) controller to achieve desired control performance such as stability and zero steady state error. We choose to use a P controller instead of a more sophisticated controller such as a PID (Proportional-Integral-Derivative) controller because the actuator $f_i(k) = d_i(k) + f_i(k-1)$ already includes an integrator such that zero steady state error can be achieved without resorting to an I (Integral) part. The D (Derivative) part is not used because it may amplify the noise in temperature introduced by measurement. The Z-domain form of our P controller is:

$$C(z) = \frac{1}{\Psi_i} \tag{10}$$

The transfer function of the closed-loop system controlled by controller (10) is:

$$\frac{T(z)}{D(z)} = \frac{z}{z^2 - \Theta_i z + \Theta_i} \tag{11}$$

It is easy to prove that the controlled system is stable and has zero steady state errors when the system model (9) is accurate. The detailed proofs can be found in a standard control textbook [19] and are skipped due to space limitations.

4.3 Control Analysis for Model Variation

In this subsection, we analyze the system stability when the system model (9) varies for different processors. A fundamental benefit of the control-theoretic approach is that it gives us theoretical confidence for system stability, even when the controller is used in a different working condition.

A different processor usually has different thermal resistance and capacitance R_j and C_j, where $R_j \neq R_i$ and $C_j \neq C_i$. Therefore, even though the designed P controller in (10) is proven to be stable on the processor used to derive the nominal system model (9) by system identification, the system stability when the controller is used on a different processor must be theoretically reevaluated.

We now outline the detailed steps to analyze the stability when the system model changes for different processors.

1. We first get the actual system model of a different processor by conducting automated system identification on the processor. Since the value of Θ_i is determined by the thermal resistance and capacitance R_i and C_i, Θ_i will be different for a different processor. Therefore, the actual system model is in the following format:

$$t_i(k) = (1+g\Theta_i)t_i(k-1) - g\Theta_i t_i(k-2) + \Psi_i d_i(k-1) \tag{12}$$

where $g\Theta_i$ is the actual parameters that may be different from Θ_i in the nominal model (9).

2. The controller function $C(z)$ presented in (10) represents the control decision made based on the nominal model (9). We then derive the closed-loop system transfer function by plugging the controller into the actual system. The closed-loop transfer function represents the system response when the controller is applied to a system whose model is different from the one used to design the controller. The closed-loop transfer function is:

$$\frac{T(z)}{D(z)} = \frac{z}{z^2 - g\Theta_i z + g\Theta_i} \tag{13}$$

3. Finally, we derive the stability condition of the closed-loop system (13). According to control theory, the closed-loop system is stable if all the poles of (13) locate inside the unit circle in the complex space. The poles are calculated as the roots of the denominator in (13), *i.e.*, the following equation:

$$z^2 - g\Theta_i z + g\Theta_i = 0 \tag{14}$$

The stability condition of applying the controller designed based on the nominal model (9) to a processor with a different system model can be stated as: if the roots of (14) all locate inside the unit circle in the complex space, the controlled system is stable. We have developed a script to analyze system stability automatically using numerical methods.

5 Coordination Analysis

We now analyze the coordination needed for the utilization and thermal control loops to work together. A major contribution of this paper is to demonstrate the importance of a novel methodology for coordinating different control loops. Both the utilization and thermal control loops have been proven to be stable in previous sections. If both the two control loops are still stable under the impact from the other loop, the entire system is stable.

The analysis of the impact of one loop on the other loop is similar to the stability analysis of a control loop with an actual system model that is different from its nominal model (as in Section 4.3). If the actual model is known, we can analyze stability by examining whether all the poles of the closed-loop system locate inside the unit circle in the complex space. However, in a real DRE system, the actual system model may vary significantly at runtime in an unpredictable way. Therefore, we adopt robust control theory to derive the stability condition for a given DRE system. The differences or errors between the actual system model and the nominal model are referred to as *uncertainty* in robust control theory. The main advantage of robust control is that uncertainty is considered explicitly in the stability analysis of a feedback control system. This characteristic makes robust control well suitable for analyzing the stability of a control loop when it is under the impact from another loop.

We now analyze the stability of the utilization control loop under the impact from the thermal controller using small gain theorem [12]. The theorem gives stability condition of a *unified* block diagram with three transfer functions called feed-forward matrix $K(z)$, sensitivity function $S(z)$, and Δ, which is explained below. The general steps of applying the small gain theorem to the utilization control loop are as follows:

1. Derive feed-forward matrix $K(z)$ and sensitivity function $S(z)$ of the utilization control loop. The matrices are in the following forms:

$$\begin{aligned} K(z) &= \frac{zK_{mpc}}{z-1}[I + \frac{\Upsilon K_{mpc}}{z-1}]^{-1} \\ S(z) &= [I + P_0(z)K(z)\Psi]^{-1} \end{aligned}$$

   where $P_0(z)$ is the transfer function of (3) when $\mathbf{G}$ is the identity matrix. K_{mpc}, Υ, and Ψ are three parameters of the MPC controller. In Section 3, the MPC controller is formulated as an optimization problem. To derive the three parameters, it is necessary to transform the optimization formulation to its block diagram representation using the methods in [21].

2. Transform the system model (3) to the form: $P(z) = P_0(z) + \Delta$. Δ represents the difference between $P_0(z)$ and the actual transfer function with the impact of the thermal loop. To derive the stability condition, we need to know the maximum value of the frequency response of Δ, which is denoted as $\|\Delta\|_\infty$. The value can be computed using Matlab command `fitmag` given the range of DVFS levels available for the thermal controller to throttle the processor.

3. Given $K(z)$, $S(z)$, and $\|\Delta\|_\infty$ from steps 1 and 2, derive the robust stability condition for the utilization control loop by applying the small gain theorem. The theorem states that the closed-loop system is stable under the uncertainty Δ if and only if

$$\bar{\sigma}[K(\omega)S(\omega)]\|\Delta\|_\infty < 1. \qquad (15)$$

   where ω is the angular frequency, i.e., $\omega = 2\pi f$ and $f \in (-\infty, +\infty)$ is the Fourier transform variable. $\bar{\sigma}$ is the maximum singular value. For example, suppose $\|\Delta\|_\infty = \gamma$, according to the theorem, $\bar{\sigma}[K(\omega)S(\omega)]$, which equals the maximum possible amplification of $K(\omega)S(\omega)$ for all frequencies, must be less than $\frac{1}{\gamma}$. If the inequation (15) does not hold, we can adjust parameters mentioned in steps 1 and 2 to reduce the maximum possible amplification of $K(\omega)S(\omega)$ or narrow the limited range of physical DVFS levels to reduce $\|\Delta\|_\infty$.

Stability analysis of the thermal control loop under the impact from the utilization controller is similar and not presented due to space limitations.

We have performed the above coordination analysis procedures for the DRE system deployed on our testbed and evaluated in our experiments. Our results show that both the two control loops are stable even under the impacts from each other, and thus the entire DRE system is stable. If a system is found to be unstable, workload and platform reconfigurations can be tuned to adjust the system for stability according to derived robust stability conditions.

6 System Implementation

In this section, we introduce our testbed, workload, and the implementation details of the two control loops.

6.1 Testbed and Workload

Our testbed includes four Linux servers (RTES1 to RTES4) running end-to-end real-time tasks and a desktop machine running the cluster level utilization controller. Four servers are equipped with 2.4GHz AMD Athlon 64 3800+ processors with 1GB RAM and 512KB L2 Cache. The desktop machine is a Dell OptiPlex GX520 with 3.00GHz Intel Pentium D Processor and 1GB RAM. All machines are connected by a 100Mbps Ethernet switch. The controller machine runs Windows XP while all other servers run openSUSE 11 and the Linux kernel is 2.6.25 with real-time support.

We implement our control architecture in FC-ORB, an open-source real-time Object Request Broker (ORB) middleware system [20]. FC-ORB supports end-to-end real-time tasks based on the end-to-end scheduling framework [13]. FC-ORB implements the release guard protocol to enforce the precedence constraints among subtasks.

Our experiments run a medium-sized workload that comprises 12 end-to-end tasks (with a total of 25 subtasks). The subtasks on each processor are scheduled by the RMS algorithm [13]. Each task's end-to-end deadline is $d_i = n_i/r_i(k)$, where n_i is the number of subtasks in task T_i and $r_i(k)$ is the current rate of T_i. Each end-to-end deadline is evenly divided into subdeadlines for its subtasks. The resultant subdeadline of each subtask T_{ij} equals its period, $1/r_i(k)$. The utilization set point of every processor is set to its RMS schedulable utilization bound [13], i.e., $B_i = n_i(2^{1/n_i} - 1)$, where n_i is the number of subtasks on P_i. All (sub)tasks meet their (sub)deadlines if the desired utilization on every processor is enforced.

6.2 Control Components

We now introduce the implementation details of each component in the coordinated solution.

Utilization Monitor: The utilization monitor uses the `/proc/stat` file in Linux to estimate the CPU utilization in each sampling period. The `/proc/stat` file records the number of *jiffies* (usually 10ms in Linux) when the CPU is in user mode, user mode with low priority (nice), system mode, and when used by the idle task, since the system starts. At the end of each control period, the utilization monitor reads the counters, and estimates the CPU utilization as 1 minus the number of jiffies used by the idle task in the last control period and then divided by the total number of jiffies in the same period.

Utilization Controller: The controller is implemented as a single-thread process running separately on the desktop machine. Each time its periodic timer fires, the controller sends utilization requests to all processors in the cluster. The incoming replies are handled asynchronously so that the controller can avoid being blocked by an overloaded processor. After the controller collects replies from all processors, it executes the control algorithm introduced in Section 3 to compute new task rates. The controller then sends the tasks' new rates to the rate modulators on processors for enforcement. If a processor does not reply in an entire control period, its utilization is treated as 100%, as the controller assumes this processor is overloaded with its (sub)tasks and so cannot respond. The control period of the utilization loop is 4 seconds.

Rate Modulator: A Rate Modulator is located on each processor. It receives the new rates from the controller and then resets the timer interval of the first subtask of each task whose invocation rate has been changed.

Temperature Monitor: AMD processors have built-in circuits to measure the chip temperature. Two most common types of circuits are thermal diode and on-die digital thermometers. The thermal diodes are normally placed close to the maximum temperature spots (*i.e.*, hot spots) of an AMD chip [22]. The thermal values can be accessed via the Machine Specific Register (MSR) by user-mode applications. In this paper, we use the utility functions from the *lm-sensors* project [23], which provide a uniform user interface to monitor a wide range of processors.

Thermal Controller: The controller is implemented as a single-thread process running on each processor. With a control period of 4 seconds, the controller periodically reads the temperature of the processor, executes the control algorithm presented in Section 4.2 to compute the desired CPU frequency, and sends the new frequency to the frequency modulator on the processor.

Frequency Modulator: We use AMD's Cool'n'Quiet technology to enforce the new CPU frequency. AMD Athlon 64 3800+ microprocessor has 5 discrete CPU frequency levels. To change CPU frequency, one needs to install the *cpufreq* package and then use root privilege to write the new frequency level into the system file */sys/devices/system/cpu/cpu0/cpufreq/scaling_setspeed*. A routine periodically checks this file and resets the CPU frequency accordingly. The average overhead (*i.e.*, transition latency) to change frequency in AMD Athlon processors is about $100\mu s$ according to the AMD white paper report.

Since the new CPU frequency level periodically received from the proportional controller could be any value that is not exactly one of the five supported frequency levels. Therefore, the modulator code must locally resolve the output value of the controller to a series of supported frequency levels to approximate the desired value. For example, to approximate 2.89GHz during a control period, the modulator would output the sequence 2.67, 3, 3, 2.67, 3, 3, etc on a smaller timescale. To do this, we implement a first-order delta-sigma modulator, which is commonly used in analog-to-digital signal conversion. The detailed algorithm of the first-order delta-sigma modulator can be found in [24].

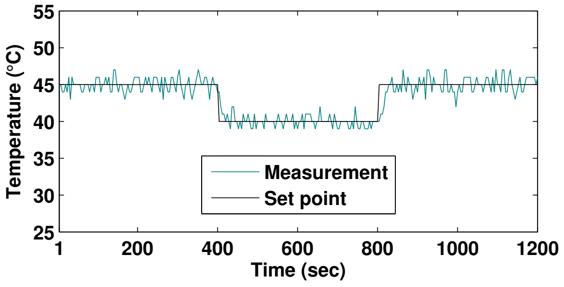


Figure 3. Thermal Controller

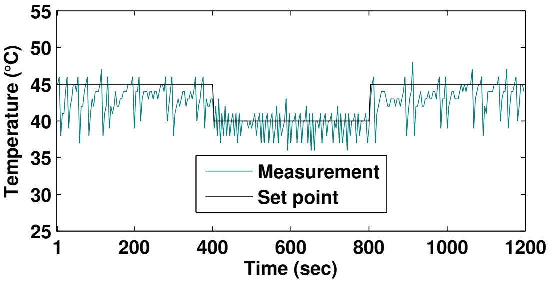


Figure 4. Ad Hoc

7 Empirical Results

In this section, we first evaluate the thermal controller alone by comparing it with a commonly used ad hoc solution. We then test the coordinated control solution in the case of thermal variations on a single processor and on all the processors. Finally, we examine the coordinated solution under task execution time variations.

We use two baselines for comparison in this paper. OPEN is a typical open-loop solution that configures the task rates and processor DVFS levels in a static way. While OPEN can initially achieve the desired CPU utilizations and processor temperatures, OPEN may fail when task execution times or system conditions dynamically change at runtime. Ad Hoc represents a commonly used solution to thermal control of a processor. When the current processor temperature is lower than the set point, Ad Hoc will increase the processor's DVFS level by one. When the temperature is lower than the set point, Ad Hoc sets the DVFS level to the lowest one to avoid overheating. A fundamental difference between Ad Hoc and our thermal controller is that Ad Hoc simply raises the DVFS level by one step or sets it to the lowest level, depending on whether the measured temperature is lower or higher than the set point. In contrast, our thermal controller computes a fractional DVFS level based on well-established control theory and uses the frequency modulator to approximate this output with a series of discrete DVFS levels.

7.1 Thermal Controller

In this experiment, we disable the utilization control loop to evaluate the performance of the thermal controller on RTES1. The temperature set point is initially 45°C in our experiment. Between time 400s and 800s, we reduce the set point to 40°C to emulate a thermal emergency event. As shown in Figure 3, under our thermal controller, the measured processor temperature converges to the desired level

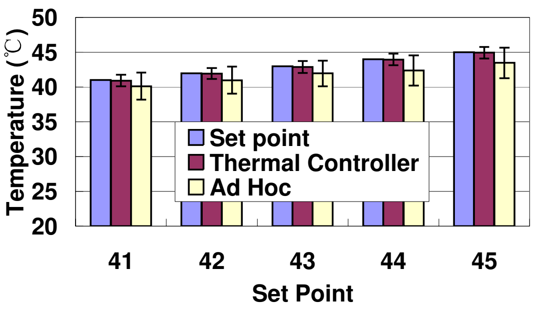


Figure 5. Comparison of thermal controller and Ad Hoc under different temperature set points

promptly after the set point is changed. Despite the measurement noise from the processor thermometer, the thermal controller allows the temperature to stay very close to the set point by dynamically throttling the processor DVFS level. In contrast, Figure 4 shows that Ad Hoc causes the processor temperature to oscillate dramatically because Ad Hoc simply raises the DVFS level by one step or sets it to the lowest level. As neither of the two temperature set points (i.e., 45°C and 40°C) can be exactly achieved by the processor by staying at any of the several available DVFS levels, Ad Hoc has to continuously throttle the processor DVFS level around a set point. As a result, the average processor temperature under Ad Hoc cannot settle to the set point, leading to a steady-state error, as shown in Figure 4.

Figure 5 compares the processor temperature achieved by the thermal controller and Ad Hoc under different set points from 41°C to 45°C. The average temperatures and the standard deviations are calculated based on the measured temperature readings when the controllers enter their steady states. The thermal controller has much smaller steady-state errors and also smaller deviations compared to Ad Hoc. Note that the smaller steady-state errors can contribute to higher processor frequencies and thus better system performance (e.g., higher task rates). In addition, if the thermal controller is given an unreasonably high set point, the P controller will saturate at the highest DVFS level and thus allow the system to run at its peak performance. The experiments demonstrate that our thermal controller designed based on control theory outperforms a commonly used thermal control solution by having more accurate thermal control.

7.2 Thermal Variations

In this experiment, we enable both the utilization controller and the thermal controller to examine the simultaneous thermal and timeliness guarantees provided by the coordinated control solution in two scenarios.

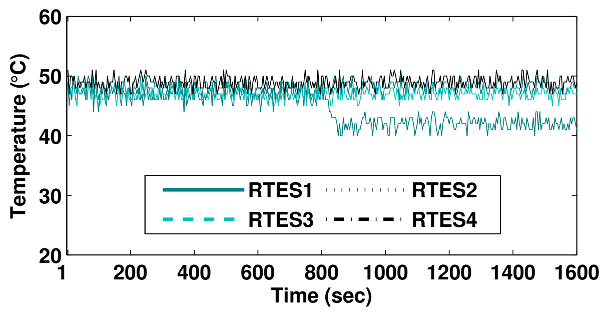
In the first scenario, the temperature set point of a single processor is changed from 47°C to 42°C at 800s to emulate a local thermal emergency event. Although the experiment in Section 7.1 has shown that our thermal controller can achieve the desired new temperature set point by conducting DVFS, the lowered CPU frequency may increase the execution times of the real-time tasks in the system and thus cause deadline misses if there is no control for task timeliness. Figure 6(a) shows that the thermal controller on RTES1 can precisely achieve the desired new temperature set point by reducing the CPU frequency of RTES1. As a result, Figure 6(b) shows that the CPU utilization of RTES1 rises to nearly 100% and so violates the schedulable utilization bound. With effetive utilization control, the coordinated control solution reduces the task rates to lower the CPU utilization of RTES1 to the desired set point. Other processors in the system also have small utilization variations because rate adaptation of a task affects all its subtasks running on multiple processors.

In the second scenario, the temperatures of all the processors are changed from 47°C to 42°C at 800s to emulate a global thermal emergency event. Figure 7(a) shows that the thermal controllers on all the processors successfully lower their processor temperatures to the new set point by reducing CPU frequencies of the processors. As a result, Figure 7(b) shows that all the processors have significant increases of CPU utilization. The coordinated control solution adjusts the rates of the end-to-end tasks in the system at the cluster level to lower the CPU utilizations of all the processors to the desired set point. The two experiments demonstrate that the coordinated control solution can provide simultaneous thermal and timeliness guarantees when the system has either a local or a global thermal emergency event.

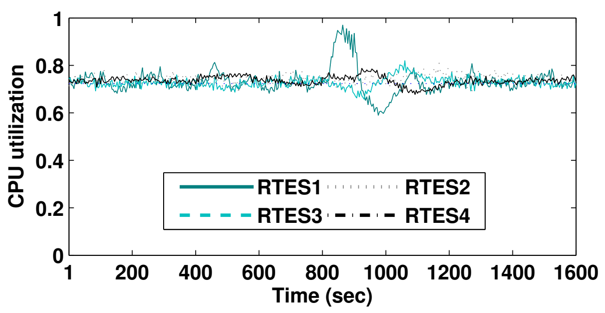
7.3 Task Execution Time Variations

In this experiment, we examine the simultaneous thermal and timeliness guarantees provided by the coordinated control solution when the execution times of the subtasks on RTES1 have a 25% of increase at 600s. Unpredictable execution time increases may cause the system to violate its utilization bounds, resulting in deadline misses. In addition, execution time variations may also increase system temperature and cause thermal emergency if the processor stays overloaded for a long time.

We first examine the performance of OPEN, which configures task rates and processor DVFS levels in a static way. While OPEN can initially achieve the desired utilizations and temperatures, Figure 8(a) shows the utilization of RTES1 increases to 95% at 600s and stays above the utilization bound in the rest of the run. Figure 9 shows that the temperature of RTES1 is consequently higher than the desired set point. Due to the lack of adaptation, OPEN may cause system malfunctions and may even reduce the lifetime of the processor. In constrast, Figure 8(b) shows that the coordinated solution can effectively control the increased CPU utilizaton by conducting rate adapation. Figure 9 shows that the processor temperature of RTES1 has an instantaneous increase at 620s in response to the execution time increase at 600s. However, the coordinated solution immediately reduces the temperature to the desired set point by throttling CPU frequency. This experiment demonstrates that the coordinated control solution can provide simultaneous thermal and timeliness guarantees when task execution times vary at runtime.

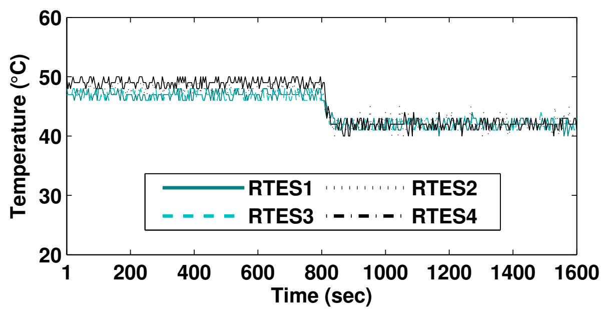


(a) Temperatures of the four processors

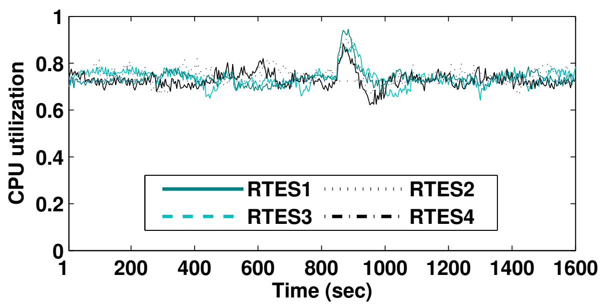


(b) CPU utilizations of the four processors

Figure 6. Thermal variation on a single processor (RTES1)

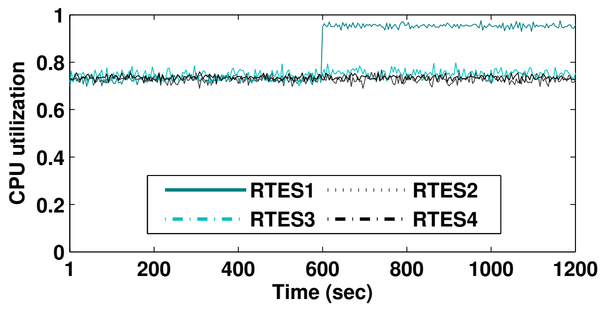


(a) Temperatures of the four processors

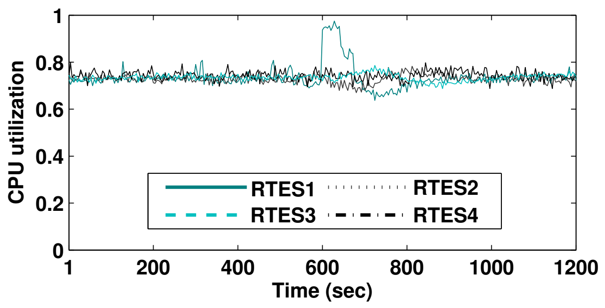


(b) CPU utilizations of the four processors

Figure 7. Thermal variations on all the four processors



(a) OPEN



(b) Coordinated control solution

Figure 8. Variations of task execution times on a single processor (RTES1)

8 Related Work

Some related work has been done to manage power or energy for real-time systems (e.g., [25][26]). In this paper, we try to explicitly control the processor temperatures for DRE systems. Previous research on thermal management focuses mainly on general computer systems. For example, Brooks et al. [10] propose a dynamic thermal management scheme based on heuristics. Skadron et al. [16] present several DTM schemes including a control-theoretic algorithm. Donald et al. [27] develop a control-theoretic thermal management approach for multi-core processors. However, all the aforementioned work cannot provide timeliness guarantees for real-time systems.

Several studies have proposed thermal management algorithms for real-time systems. Bansal et al. [28][29] present online algorithms to solve real-time scheduling problems while guaranteeing thermal constraints. Chen et al. [8][30] design real-time scheduling algorithms with reactive CPU speed assignment and develop several optimization algorithms to minimize the maximum system temperature in a static way. Different from their work that relies on heuristics or optimizations, we propose a coordinated solution based on control theory to provide simultaneous thermal and timeliness guarantees despite various runtime thermal and execution time variations.

Control-theoretic techniques have been applied to many computing systems. For example, various CPU utilization control algorithms (e.g., [4][2][3][20]) have been recently proposed to guarantee real-time deadlines. However, those algorithms cannot provide thermal guarantees. Recently, coordinated control solutions have been proposed for power/energy management. For example, Raghavendra et al. [17] propose a multi-layer controller for data center power management. Heo et al. [31] study the incompatibilities

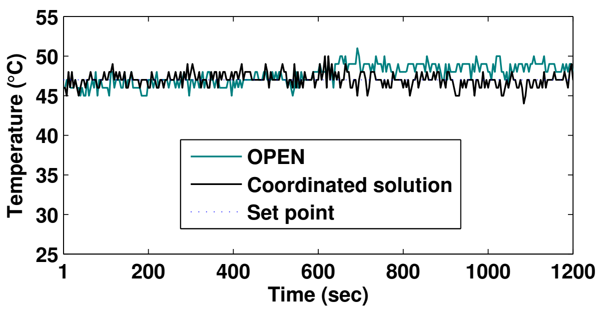


Figure 9. Comparison of system temperature under the coordinated control solution and OPEN

problems of conflicting control systems and propose a formal methodology to analyze conflicts. Another coordination strategy has been proposed in [11] by forcing different control loops to run on different timescales. In contrast, our solution is designed based on robust control theory to allow the thermal and utilization control loops to run on their respective desired timescales for prompt control actions and simultaneous guarantees.

9 Conclusions

Today's DRE systems face an increasing probability of overheating and even thermal failures, due to their continuously decreasing feature size and increasing demand for computation capabilities. As a result, their temperature must be explicitly controlled for improved reliability. However, existing work provides either real-time guarantees or thermal management in an isolated manner. In this paper, we have presented a coordinated control solution that can provide simultaneous thermal and timeliness guarantees for real-time embedded systems running in unpredictable environments. The thermal control loop locally controls the temperature of each processor, while the utilization control loop provides end-to-end timeliness guarantees at the cluster level. A novel coordination analysis method based on robust control theory has been proposed to coordinate the two control loops for theoretically guaranteed global system stability. Empirical results on a physical testbed demonstrate the efficacy of our control solution.

References

[1] D. Henriksson and T. Olsson, "Maximizing the use of computational resources in multi-camera feedback control," in *RTAS*, 2004.

[2] C. Lu, X. Wang, and X. Koutsoukos, "Feedback utilization control in distributed real-time systems with end-to-end tasks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 16, no. 6, 2005.

[3] X. Wang, D. Jia, C. Lu, and X. Koutsoukos, "DEUCON: Decentralized end-to-end utilization control for distributed real-time systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 7, 2007.

[4] C. Lu, X. Wang, and C. Gill, "Feedback control real-time scheduling in ORB middleware," in *RTAS*, 2003.

[5] L.-T. Yeh and R. C. Chu, *Thermal Management of Microelectronic Equipment: Heat Transfer Theory, Analysis Methods, and Design Practices*. ASME Press, 2002.

[6] R. Viswanath, V. Wakharkar, A. Watwe, and V. Lebonheur, "Thermal performance challenges from silicon to systems," *Intel Technology Journal*, vol. 23, no. 3, 2000.

[7] D. Anderson, J. Dykes, and E. Riedel, "More than an interface—SCSI vs. ATA," in *FAST*, 2003.

[8] J.-J. Chen, S. Wang, and L. Thiele, "Proactive speed scheduling for frame-based real-time tasks under thermal constraints," in *RTAS*, 2009.

[9] S. Wang and R. Bettati, "Reactive speed control in temperature-constrained real-time systems," *Real-Time Systems Journal*, vol. 39, no. 1-3, 2008.

[10] D. Brooks and M. Martonosi, "Dynamic thermal management for high-performance microprocessors," in *HPCA*, 2001.

[11] X. Wang, X. Fu, X. Liu, and Z. Gu, "Power-aware cpu utilization control for distributed real-time systems," in *RTAS*, 2009.

[12] K. Zhou, J. Doyle, and K. Glover, *Robust and Optimal Control*. Prentice Hall, 1996.

[13] J. W. S. Liu, *Real-Time Systems*. Prentice Hall, 2000.

[14] J. Sun and J. Liu, "Synchronization protocols in distributed real-time systems," in *ICDCS*, 1996.

[15] K. Skadron, T. Abdelzaher, and M. R. Stan, "Control-theoretic techniques and thermal-RC modeling for accurate and localized dynamic thermal management," in *HPCA*, 2002.

[16] K. Skadron, M. R. Stan, K. Sankaranarayanan, W. Huang, S. Velusamy, and D. Tarjan, "Temperature-aware microarchitecture: Modeling and implementation," *ACM Transactions on Architecture and Code Optimization*, vol. 1, no. 1, 2004.

[17] R. Raghavendra, P. Ranganathan, V. Talwar, Z. Wang, and X. Zhu, "No power struggles: Coordinated multi-level power management for the data center," in *ASPLOS*, 2008.

[18] X. Wang and M. Chen, "Cluster-level feedback power control for performance optimization," in *HPCA*, 2008.

[19] G. F. Franklin, J. D. Powell, and M. L. Workman, *Digital Control of Dynamic Systems (3rd Edition)*. Prentice Hall, 1997.

[20] X. Wang, Y. Chen, C. Lu, and X. Koutsoukos, "FC-ORB: A robust distributed real-time embedded middleware with end-to-end utilization control," *Journal of Systems and Software*, vol. 80, no. 7, 2007.

[21] J. M. Maciejowski, *Predictive Control with Constraints*. Prentice Hall, 2002.

[22] D. Li, H.-C. Chang, H. K. Pyla, and K. W. Cameron, "System-level, thermal-aware, fully-loaded process scheduling." in *IPDPS*, 2008.

[23] "Linux hardware monitoring," http://www.lm-sensors.org/.

[24] C. Lefurgy, X. Wang, and M. Ware, "Power capping: a prelude to power shifting," *Cluster Computing*, vol. 11, no. 2, 2008.

[25] H. Cheng and S. Goddard, "Online energy-aware i/o device scheduling for hard real-time systems," in *DATE*, 2006.

[26] A. Dudani, F. Mueller, and Y. Zhu, "Energy-conserving feedback edf scheduling for embedded systems with real-time constraints," in *LCTES*, 2002.

[27] J. Donald and M. Martonosi, "Techniques for multicore thermal management:classification and new exploration," in *ISCA*, 2006.

[28] N. Bansal, T. Kimbrel, and K. Pruhs, "Dynamic speed scaling to manage energy and temperature," in *FOCS*, 2004.

[29] N. Bansal and K. Pruhs, "Speed scaling to manage temperature," in *STACS*, 2005.

[30] J.-J. Chen, C.-M. Hung, and T.-W. Kuo, "On the minimization of the instantaneous temperature for periodic real-time tasks," in *RTAS*, 2007.

[31] J. Heo, D. Henriksson, X. Liu, and T. Abdelzaher, "Integrating adaptive components: An emerging challenge in performance-adaptive systems and a server farm case-study," in *RTSS*, 2007.