Feedback Thermal Control for Real-time Systems

Yong Fu*, Nicholas Kottenstette†, Yingming Chen*, Chenyang Lu*, Xenofon D. Koutsoukos†, Hongan Wang‡
*Dept. of CSE, Washington University, St. Louis, MO, {fuy,yingming,lu}@cse.wustl.edu
†Dept. of EECS, Vanderbilt University, Nashville, TN, {nkottens,Xenofon.Koutsoukos}@vanderbilt.edu
‡ Institute of Software, Chinese Academy of Sciences, wha@iel.iscas.ac.cn

***Abstract*—Thermal control is crucial to real-time systems as excessive processor temperature can cause system failure or unacceptable performance degradation due to hardware throttling. Real-time systems face significant challenges in thermal management as they must avoid processor overheating while still delivering desired real-time performance. Furthermore, many real-time systems must handle a broad range of uncertainties in system and environmental conditions. To address these challenges, this paper presents *Thermal Control under Utilization Bound (TCUB)*, a novel thermal control algorithm specifically designed for real-time systems. TCUB employs a nested feedback loop that dynamically controls both processor temperature and CPU utilization through task rate adaptation. Rigorously modeled and designed based on control theory, TCUB can maintain both desired processor temperature and CPU utilization, thereby avoiding processor overheating and maintaining desired soft real-time performance. A salient feature of TCUB lies on its capability to handle a broad range of uncertainties in terms of processor power consumption, task execution times, ambient temperature, and unexpected thermal faults. The robustness of TCUB makes it particularly suitable for real-time embedded systems that must operate in highly unpredictable environments. The advantages of TCUB are demonstrated through extensive simulations under a broad range of system and environmental uncertainties.**

***Index Terms*—real-time systems; thermal control; utilization control; anti-windup control;**

I. Introduction

Real-time embedded systems face significant challenges in thermal management as they adopt modern processors with increasing power density and compact architecture. Such systems must avoid processor overheating while still maintaining desired real-time performance. While modern processors usually rely on hardware throttling mechanisms to prevent overheating, such mechanisms cause performance degradation unacceptable for real-time applications.

Moreover, real-time embedded systems must deal with a broad range of uncertainties in system characteristics and environmental conditions:

- *Power consumption*: The power consumption of a processor may vary significantly when running different tasks, and can be influenced by the instructions executed [1].
- *Ambient temperature*: In contrast to servers operating in air-conditioned environments, real-time embedded systems may operate in diverse environments under a wide range of ambient temperature.
- *Thermal faults*: Due to their harsh operating conditions embedded systems can be particularly susceptible to failures of cooling subsystems [2].
- *Task execution times*: The execution times of many real-time applications are unknown a priory because their executions are strongly influenced by the operating environment and sensor inputs.

To meet these challenges, we present *Thermal Control under Utilization Bound (TCUB)*, a novel dynamic thermal management algorithm specifically designed for real-time embedded systems. TCUB employs feedback control loops to control *both* the processor temperature and CPU utilization by adjusting task rates. In contrast to earlier research on feedback control real-time scheduling that ignores thermal issues [3], TCUB can maintain *both* desired processor temperature and CPU utilization bound, thereby avoiding processor overheating and maintaining desired real-time performance. TCUB has the following salient features.

- TCUB features a nested feedback control structure consisting of (1) a low-rate thermal controller dealing with the slower thermal dynamics, and (2) a high-rate utilization controller handling the faster CPU utilization dynamics caused by uncertainties in task execution times. The thermal controller outputs a set-point for the CPU utilization that accounts for the thermal dynamics and is consistent with the schedulability bounds of the real-time system. This set-point is, in turn, used by the utilization controller to adjust the task rates. The modular control structure allows separate control designs optimized for thermal-protection and utilization-regulation.
- In contrast to earlier research on thermal-ware real-time scheduling that relies on accurate system and task models [4]–[8], TCUB is a highly *robust* algorithm that can handle a broad range of uncertainties in terms of processor power consumption, task execution times, thermal faults, and ambient temperature. The robustness of TCUB makes it particularly suitable for real-time embedded systems that operate in highly unpredictable environments.
- In contrast to model predictive control adopted by earlier research [9] that results in complex robustness analysis, conservative design, and incurs high computational overhead, TCUB features a simple and efficient thermal controller that integrates a discrete-time-proportional-integral-controller and a *traditional anti-windup controller* designed to enforce the desired CPU utilization bound, which has $\mathcal{O}(1)$ time complexity. The anti-windup controller is necessary to handle the schedulability bounds that impose hard saturation constraints on the output of the thermal controller (utilization set-point). Moreover, the control approach allows rigorous analysis of stability and robustness under uncertainties.

- Extensive simulation results demonstrate the stability and robustness of TCUB under a *wide range of uncertainty and operating conditions* including varying power consumption and ambient temperature, as well as thermal faults.

The rest of the paper is organized as follows. Section II-A presents a difference equation model that characterizes the thermal dynamics of real-time systems. Section IV details the design and stability analysis of TCUB. Section V provides simulation results. Section VI introduces related work. Section VII concludes the paper.

II. Problem Formulation

In this section we first present the system model adopted in this work , and then we discuss the goals of thermal control for real-time systems.

A. System Models

A key feature of our system model is that it characterizes the *uncertainties* in real-time systems in terms of task execution time, power consumption, ambient temperature, and thermal faults. We assume a single processor real-time system running n independent, periodic real-time tasks, $\{T_i | 1 \leq i \leq n\}$. Each task T_i has a period p_i. The task rate r_i of the task T_i is defined as $r_i = \frac{1}{p_i}$. Each task has a soft dealine related to its period and an estimated execution time c_i known at design time. However, the actual execution time a_i at run time is unknown and may deviate from c_i.

The rate r_i of the task T_i can be dynamically adjusted within a range $[R_{\min,i}, R_{\max,i}]$. Earlier work had shown that task rates in many real-time applications (e.g., digital feedback control [10] and multimedia [11]) can be adjusted in certain ranges without causing system failure. A task running at a higher rate contributes a higher value to the application at the cost of higher CPU utilization.

When tasks are running on the processor, the active power consumed by the processor fluctuates significantly. Earlier work refers to such significant power variation during run time as power phase behavior [1]. At the instruction level, different instruction types, inter-instruction overhead, memory system state, and pipeline related effects cause power fluctuation [12]. Therefore, while the *estimated* active power of the processor, P_a, is known, the actual active power of the processor may deviate from the estimate at run time. When the processor is idle, the processor consumes idle power P_{idle}.

We adopt the well known thermal RC model to characterize the thermal dynamics of the processor [2], [13]:

$$\frac{dT(t)}{dt} = -b_2(T(t) - T_0) + b_1 P(t) \tag{1}$$

where $T(t)$ is the temperature of the processor, T_0 is ambient temperature, $P(t)$ is the actual power consumed by the processor, $b_1 = \frac{1}{C_{\mathsf{th}}}$ and $b_2 = \frac{1}{R_{\mathsf{th}} C_{\mathsf{th}}}$, where C_{th} is heat capacity and R_{th} is heat resistance. As embedded systems may operate in diverse environments, the ambient temperature T_0 may change. Moreover, thermal faults (e.g., fan failure) may cause significant change to the thermal resistance [2]. A thermal control algorithm designed for real-time systems must handle these uncertainties at run time.

B. Design Goals

Our thermal control algorithm is designed to meet two primary requirements: (1) to prevent processor overheating, and (2) to maintain desired soft real-time performance. Due to the uncertainties faced by real-time systems, TCUB adopts a feedback control approach that dynamically controls the processor temperature and real-time performance. It allows users to specify a temperature set-point T_R, and a utilization bound $U_{\max}$. For processors support hardware throttling, the temperature set-point is below the temperature threshold for hardware throttling so as to avoid the unpredictable performance degradation caused by throttling. For processors that do not support throttling, the temperature set-point should be below the maximum temperature acceptable to the processor. The CPU utilization bound $U_{\max}$ should be below the schedulable utilization bound of the real-time scheduling policy (e.g., [14]).

TCUB is designed to prevent processor overheating by keeping the temperature below or close to the temperature set-point T_R, and to maintain desired software real-time performance by enforcing the CPU utilization bound $U_{\max}$.[1] Moreover, TCUB must handle uncertainties in terms of power consumption, task execution times, ambient temperature, and thermal fault. Finally, the control algorithm should be simple and efficient to provide a practical solution for resource-limited embedded systems.

III. Overview of TCUB

We propose a multi-rate nested feedback-loop control approach to manage both the temperature and the utilization. As shown in Fig. 1, there are two control loops in TCUB that operate at different time scales. The outer loop is responsible for thermal control and runs at a lower rate than the inner loop responsible for utilization control. In the outer loop the thermal controller aims to enforce the specified temperature set-point T_R. At the end of the k^{th} sampling period of the outer loop, the thermal controller computes the utilization set-point $U_s(k)$ for the utilization controller of the inner loop based on the measured temperature $T(k)$ provided by the thermal monitor. The inner-loop utilization controller ensures that the utilization converges to the set-point $U_s(k)$ computed by the thermal controller by adjusting the task rates. At the k'_{th} sampling period of the inner loop, the utilization controller output the task rate change $\Delta r(k')$ based on the measured utilization $U(k')$. The rate actuator adjusts tasks rate based on the output of the utilization controller. Our multi-rate nested control approach has several important advantages.

1) The thermal dynamics are typically significantly slower than the utilization dynamics, which motivates a multi-rate control approach. The processor thermal-control problem usually involves a *large* thermal time-constant ($\tau_{\mathsf{th}} = R_{\mathsf{th}} C_{\mathsf{th}} \approx 150$ seconds) whereas existing utilization controllers (which we incorporate into our design) typically have dynamic responses within a few seconds (e.g., less than 4 seconds [3]).

[1]As TCUB only controls the average CPU utilization dynamically, it is not suitable for hard real-time systems.

2) Unlike computationally intensive model predictive control adopted by earlier work on thermal control [9], our proposed nested control architecture greatly simplifies the control algorithms. It requiring neither complicated gain-scheduling tables nor complicated on-line optimization algorithms. The lower rate thermal-control loop further reduces computational burden.
3) We provide a stability and robustness analysis for the thermal-controller, based on the necessary and sufficient Nyquist Stability criterion which allows us to *directly* relate uncertain physical properties of our thermal-dynamic control problem, whereas the model predictive control approach [9] has to rely on a *conservative* small gain assumption and offers little insight into the physical parameter uncertainties which directly affect stability and performance.

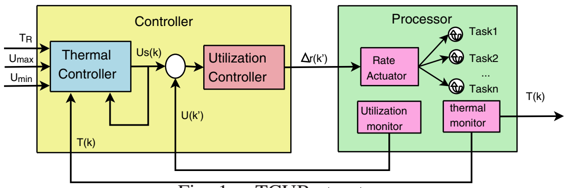


Fig. 1. TCUB structure

Specifically, the nested control loops work as follows. The thermal and utilization controller employ two sampling periods: T_s, which denotes the sampling period of the processor's temperature; and T_u, which is the sampling period of the utilization ($T_u < T_s$). At the end of the k^{th} temperature sampling period, the feedback loop is invoked and executes the following steps:

1) The temperature monitor sends the processor's temperature $T(k)$ to the thermal controller over the last sampling period.
2) The thermal controller calculates the CPU utilization *set-point* of the processor, $U_s(k)$, based on $T(k)$ and temperature reference. It then sends $U_s(k)$ to the utilization controller. Note $U_s(k)$ is effectively held for m samples in which m is a positive integer which relates the outer-loop sample time T_s to the inner-loop sample time T_u such that $T_s = mT_u$.
3) The utilization controller adjusts the task rates through the rate actuator at each T_u sampling period so as to track the utilization set-point $U_s(k)$. In TCUB, we employ FC-U [3] as the utilization controller. FC-U uses a Proportional controller to ensure the utilization set-point; its effectiveness in single processor real-time systems has been validated in past studies.

One benefit of our nested control structure is modular design, that is, we can design the two control loops separately. For utilization control loop we reuse the well studied feedback control utilization controller FC-U [3]. The effectiveness of FC-U is justified by the simulations and experiments. In the following sections, we only focus on the thermal controller design and stability analysis.

IV. Thermal Control Design and Analysis

The principal challenge for the thermal controller design is to guarantee that a maximum allowable temperature T_R is not exceeded while the thermal-control output $U_s(k)$ is subject to actuator-saturation which is governed by a set of utilization bounds $\{U_{\min}, U_{\max}\}(0 \leq U_{\min} < U_{\max} \leq 1)$. The maximum utilization bound $U_{\max}$ is the scheduler-dependent utilization bound beyond which tasks may miss a deadline. The minimum utilization bound $U_{\min}$ can be determined by taking the sum of the product of each *minimum achievable task execution time* with each corresponding *minimum allowable task rate* for a given system. The thermal controller is required to regulate the temperature of the processor to track T_R subject to the constraints of utilization by its output $U_s(k)$. Therefore, a proportional-integrator (PI) controller with an integrator-anti-windup controller is proposed to determine $U_s(k)$ while addressing actuator limitations in order to guarantee stability. This simple yet elegant outer-thermal control loop can be run at a significantly lower-rate without any noticeable performance loss due to the systems high thermal time constant.

In this section we describe the control design and analysis of TCUB. In the following sections we present the design of thermal controller and the stability analysis.

A. Dynamic Model for Thermal Control

As a foundation for the design of the thermal controller, we derive a discrete-time, difference-equation model that characterizes the dynamic relationship between the CPU utilization $U(k)$ (the control input) and the processor temperature $T(k)$ (the controlled variable). We first characterize the relationship between the power consumption and the CPU utilization and then derive a discrete-time model based on the thermal RC model .

First, we characterize the relationship between the power consumption of the processor and its CPU utilization. CPU utilization is the fraction of the time when CPU is active in a time interval. Let $U(k)$ denote the CPU utilization in the k^{th} sampling period. The average power of the processor in k^{th} sampling period, $\bar{P}(k)$, has the following relationship with $U(k)$:

$$\begin{aligned} \bar{P}(k) &= G_p P_a U(k) + P_{\text{idle}}(1 - U(k)) \\ &= (G_p P_a - P_{\text{idle}})U(k) + P_{\text{idle}} \end{aligned} \quad (2)$$

where G_p represents the ratio between the actual active power at run time and the estimated active power P_a. In (2) $G_p P_a$ is the actual power when the CPU is active, and $U(k)$ is the fraction of time when the CPU is active. P_{idle} is the power when the CPU is idle, and $1 - U(k)$ is the fraction time when the CPU is idle. The same power model is also used in temperature simulation of server systems [15].

Next, we transform the thermal RC model (1) to a discrete-time model. Denote the Laplace transform of $T(t)$ as $T(s)$ and $P(t)$ as $P(s)$ from (1) we have the following model

$$T(s) = \frac{R_{\text{th}}}{R_{\text{th}} C_{\text{th}} s + 1} P(s) + \frac{1}{R_{\text{th}} C_{\text{th}} s + 1} T_0. \quad (3)$$

For the thermal control analysis we need to derive a discrete-time model to *approximate* this system. The thermal controller issues a fixed-periodic utilization set-point which the inner-loop utilization controller closely and quickly regulates to. This utilization set-point is proportional to the average power consumed by the processor, as previously mentioned the thermal-time constant is large, therefore the effects of transients are *negligible*. Therefore, a ZOH-equivalent model is appropriate to approximate a discrete-time model of the thermal dynamics of the system. It is straightforward to derive the linear ZOH-equivalent discrete time model from (3) as follows [16] :

$$T(k+1) = \Phi T(k) + (1-\Phi)T_0 + R_{\mathsf{th}}(1-\Phi)P(k) \quad (4)$$

where k represents k^{th} sampling period, $\Phi = \exp(-\frac{T_s}{R_{\mathsf{th}}C_{\mathsf{th}}})$ and T_s is the sampling period.

Then we combine the thermal RC model (1) and the relationship between power and utilization (2), specifically, by substituting $P(k)$ for $\bar{P}(k)$, we could derive the model employed in thermal control:

$$\begin{aligned} T(k+1) = \Phi T(k) + R_{\mathsf{th}}(1-\Phi)(G_aP_a - P_{\mathrm{idle}})U(k) \\ + R_{\mathsf{th}}(1-\Phi)P_{\mathrm{idle}} + (1-\Phi)T_0 \quad (5) \end{aligned}$$

B. Thermal Controller Design

The structure of thermal controller we proposed is illustrated in Fig. 2. It consists of a proportional-integral (PI) controller (denoted as $K(z)$), an anti-windup controller (denoted as $\hat{H}(z)$) which is determined from the model $\hat{H}(z)$ and a saturation block. The PI controller's output is limited by the saturated block and then the utilization set-point output by the thermal controller cannot surpass the utilization bound assigned by the users. Essentially anti-windup controller transforms nonlinear behavior of the real-time systems induced by the utilization bounds to linear behavior so that normal linear control design could be exploited. The input of the PI

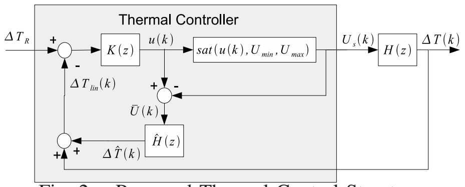


Fig. 2. Proposed Thermal Control Structure.

controller is the error between the reference trajectory and linearized temperature $\Delta T_{\mathrm{lin}}(k)$. The control output of the PI controller, $u(k)$, is limited to enforce utilization bounds by the saturated block, $U_s(k) = \mathrm{sat}(u(k), U_{\min}, U_{\max})$, in which

$$\mathrm{sat}(x, x_{\min}, x_{\max}) = \begin{cases} x_{\min}, & \text{if } x < x_{\min} \\ x_{\max}, & \text{if } x > x_{\max} \\ x, & \text{otherwise.} \end{cases}$$

In the normal case, the maximum utilization $U_{\max}$ is or less than the schedulable utilization bound of the tasks set, U_B. The error between $U(k)$ and $u(k)$, denoted as $\bar{U}(k)$, is passed through a thermal model of the processor (denoted $\hat{H}(z)$) which generates a compensation term $\Delta\hat{T}(k)$, when combined with the actual processor temperature difference $\Delta T(k)$, a *linearized* temperature difference ($\Delta T_{\mathrm{lin}}(k) = \Delta\hat{T}(k) + \Delta T(k)$) is fed-back to the controller $K(z)$ in order to guarantee stability. This compensation is also known as *anti-windup* control. It is noted that we use the thermal model of the processor as the transfer function of the processor here but without considering dynamic of the utilization controller. This is one of the benefits of nested control structure, that is, we can design the thermal and utilization controller separately. In order to describe our implementation of the thermal controller, as presented in Algorithm 1, we denote $\hat{T}_{idle}$ as an estimate of the idle temperature $T_{idle}(t)$ and $\hat{T}_o$ as either an estimate or measurement (if available) of environmental temperature T_o.

For thermal controller design, we rewrite the model (5) in a more compact form. Note that the temperature $T(k)$ depends ultimately on the environmental temperature T_0, the idle temperature component T_{idle} which depends on the idle power component P_{idle} such that $T_{\mathrm{idle}}(t) = R_{\mathsf{th}}P_{\mathrm{idle}}$, and the active power component $\Delta T(k)$, that is, $T(t) = \Delta T(t) + T_0 + T_{\mathrm{idle}}$. Then the model (5) could be rewritten as

$$\Delta T(k+1) = \Phi\Delta T(k) + \Gamma U(k) \quad (6)$$

where $\Gamma = k_pR_{\mathsf{th}}(1-\Phi)$ and $k_p = (G_aP_a - P_{\mathrm{idle}})$. In model (6) uncertainty in G_p can be expressed in terms of the following bounds on the *actual power gain* k_p such that $k_{p\,\min} \leq k_p \leq k_{p\,\max}$.

In Z-domain the model (6) can be written as follows

$$H(z) = \frac{\Delta T(z)}{U(z)} = \frac{\Gamma}{z - \Phi}. \quad (7)$$

To design the thermal controller with the proposed structure we follow two steps. First a nominal linear controller $K(z)$ ignoring the saturating limit is designed. In this work the nominal linear controller is a PI-controller, $K(s) = K_{\mathrm{P}} + K_{\mathrm{I}}\frac{s+\omega_{\mathrm{I}}}{s}$. The discrete time controller $K(z)$ is synthesized using the IPESH-transform from the continuous time controller model $K(s)$. The IPESH-transform, like the bilinear-transform, is both a passivity and stability preserving transform which can be applied to any linear-time invariant model $K(s)$ except that it will not *suffer from warping effects* and therefore closely matches the magnitude response up to the Nyquist frequency $\frac{\pi}{T_s}$ [17], [18].

Definition 1. *[17] Let $H_p(s)$ and $H_p(z)$ denote the respective continuous and discrete time transfer functions which describe a plant. Furthermore, let T_s denote the respective sample and hold time. Finally, denote $\mathcal{Z}\{F(s)\}$ as the z-transform of the sampled time series whose Laplace transform is the expression of $F(s)$, given on the same line in [19, Table 8.1 p.600]. $H_p(z)$ is generated using the following IPESH-transform*

$$H_p(z) = \frac{(z-1)^2}{T_sz}\mathcal{Z}\left\{\frac{H_p(s)}{s^2}\right\}.$$

The resulting discrete time controller is:

$$K(z) = K_{\mathrm{P}} + K_{\mathrm{I}}\left(1 + \frac{\omega_{\mathrm{I}}T_s}{2}\right)\frac{z - \frac{2-\omega_{\mathrm{I}}T_s}{2+\omega_{\mathrm{I}}T_s}}{z-1}.$$

Secondly, an anti-windup controller $\hat{H}(z)$ is designed to limit performance deterioration in the event of a control constraints being encountered.

From aforementioned thermal control design, we can present the algorithm of the thermal controller as follows:

The thermal controller related parameters used in the algorithm are explained in Section IV-C.

C. Stability Analysis

We analyze the condition of stability of the proposed control structure in this section. For a real-time system under thermal control, stability ensures that the processor temperature converges to the temperature set-point. In order to discuss stability, we recall the following definition and the Nyquist stability theorem.

Algorithm 1 Thermal Controller

```
Require: Temperature set-point, $T_R$; Utilization bounds, $U_{\min}, U_{\max}$
 1: while At the end of sampling period do
 2:     The temperature difference set-point, $\Delta T_R(k)$ is computed by
        $\Delta T_R(k) = T_R - \left(\hat{T}_0 + \hat{T}_{\text{idle}}\right)$
 3:     The linearized temperature $\Delta T_{\text{lin}}(k)$ is computed by
        $\Delta T_{\text{lin}}(k) = \Delta T_{\text{fb}}(k) + \Delta\hat{T}(k)$ in which
        $\Delta T_{\text{fb}}(k) = T(k) - \left(\hat{T}_0 + \hat{T}_{\text{idle}}\right)$
 4:     $e(k) = (\Delta T_R(k) - \Delta T_{\text{lin}}(k))$
 5:     $u(k) = u(k-1) + K_P(e(k) - e(k-1)) + K_I\left(1 + \frac{\omega_I T_s}{2}\right)\left(e(k) - \frac{2-\omega_I T_s}{2+\omega_I T_s} e(k-1)\right)$ {PI controller}
 6:     if $U_{\min} \le u(k) \le U_{\max}$ then
 7:         $U_s(k) = u(k)$
 8:     else
 9:         if $U(k) < U_{\min}$ then {Enforce $U_s(k)$ bound}
10:             $U_s(k) = U_{\min}$
11:         else $\{U(k) > U_{\max}\}$
12:             $U_s(k) = U_{\max}$
13:         end if
14:     end if
15:     $\bar{U}(k) = u(k) - U_s(k)$
16:     $\Delta\hat{T}(k+1) = \hat{\Phi}\Delta\hat{T}(k) + \hat{\Gamma}\bar{U}(k)$. {Anti-windup controller}
17: end while
```

Definition 2. *A* stable *discrete-time linear time invariant (LTI) system is one in which all poles are inside the unit circle.*

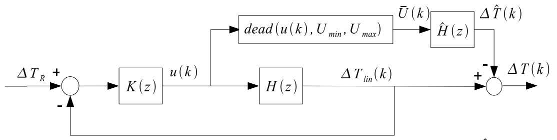


Fig. 3. Resulting feedback-structure when $H(z) = \hat{H}(z)$.

Theorem 1. *[20, p.857] Consider the closed loop consisting of $K(z)$ and $H(z)$ only depicted in Fig. 3. In order for this loop to be stable the net number of* counterclockwise *encirclements of the point -1 by the Nyquist plot of $K(e^{j\omega})H(e^{j\omega})$ as ω varies from 0 to 2π must* equal *the number of poles of $K(z)H(z)$ outside the unit circle.*

Note that Fig. 3 can be derived from Fig. 2 when $H(z) = \hat{H}(z)$. Therefore, from Theorem 1 and Fig. 3 we obtain Lemma 1 in order to verify stability of the our proposed control structure (Fig. 2).

Lemma 1. *The closed-loop system depicted in Fig. 2, in which ΔT_R is the input and ΔT_{lin} is the output, is stable if:*

i. $K(z)H(z)$ satisfy Theorem 1
ii. $\hat{H}(z) = H(z)$.

In addition, if the output $\Delta T(k)$ is to reach a steady-state output for a given input ΔT_R, then $\hat{H}(z)$ should be stable.

This leads us to the following theorem:

Theorem 2. *The closed-loop system with controller*

$$K(z) = K_P + K_I\left(1 + \frac{\omega_I T_s}{2}\right)\frac{z - \frac{2-\omega_I T_s}{2+\omega_I T_s}}{z-1}$$

depicted in Fig. 2 in which ΔT_R is the input, and ΔT_{lin} is the output is stable if:

i. $\hat{H}(z) = \frac{\hat{\Gamma}}{z-\hat{\Phi}}$, $\hat{\Gamma} \le \Gamma_{\max}$, $\hat{\Phi} \le \Phi_{\max}$
ii. $K_P = K_I = k_{GM}\frac{1+\Phi_{\max}}{2\Gamma_{\max}}$

in which $k_{GM} = 10^{-\frac{GM}{20}}$, $\Phi_{\max} = \exp(-\frac{T_s}{R_{\mathsf{th}\max}C_{\mathsf{th}}})$, $\Gamma_{\max} = k_{p\max}R_{\mathsf{th}\max}(1-\Phi_{\max})$ and $\omega_I = \frac{2(1-\Phi_{\max})}{T_s(1+\Phi_{\max})}$. where GM is the desired worst-case gain margin and , $0 \le GM < \infty$.

Due to space limit the proof is omitted here and can be found in an extended version of this paper [21] for the proof.

For our control structure shown in Fig. 2, intuitively there are only two cases to maintain stability. The first case, when the control input $U_{\min} \le u(k) \le U_{\max}$ (which implies that $\bar{U}(k) = 0$) we want to enforce stability of the *active* closed-loop system consisting of $K(z)$ and $H(z)$, and open-loop stability of $\hat{H}(z)$ (satisfied by assumption). For the second case, when the control input saturates $u(k) < U_{\min}$ or $u(k) > U_{\max}$, we want to enforce stability of the *active* closed-loop system consisting of $K(z)$ and $\hat{H}(z)$, and open-loop stability of $H(z)$ (satisfied by assumption). The anti-windup controller corresponds to the integrator component of our proposed controller $K(z)$ from deviating infinitely far from its ideal output when actuator saturation has occurred. We will always know what $U_{\max}$ will be as it is dictated by the scheduler chosen, however, some uncertainty may remain on choosing the lower-limit $U_{\min}$ due to task execution time. Therefore even choosing the ultimate lower-bound $U_{\min} = 0$ can always be a safe choice even if $U_{\min} > 0$ in that it will result in a slight sub-optimal lag in allowing the controller to increase the utilization levels due to a decrease in environmental temperature for example. Considering that environmental temperature changes are fairly slow, this slight lag is typically unnoticeable. For a more detailed discussion on anti-windup control, we refer the reader to [22], [23].

The Theorem 2 reveals the appealing feature of our thermal controller, that is, its robustness under power change and thermal fault can be guaranteed analytically. Since k_p involves uncertainty of power change represented by G_p according its definition, $k_p = (G_p P_a - P_{\text{idle}})$, $k_{p\max}$ corresponds to the maximum actual power changes that TCUB can cancel. For example, if $k_{p\max} = 510$, $P_a = 51.9w$ and $P_{\text{idle}} = 13.3w$, we can calculate that the upper limit of G_p is 10.11, that is, even if the actual power is 10.11 times by the estimated power, the thermal controller still can stabilize the system. Similarly, the capability of TCUB to handle thermal fault (modeled by increased thermal resistance) is represented by $R_{\mathsf{th}\max}$.

In addition, it is obvious that for the *steady-state case* when the $u(k) = U_s(k)$ that $\Delta T_R(k) = \Delta T_{\text{lin}}(k) = \Delta T_{\text{fb}}(k)$ due to the integrator term in $K(z)$. Therefore, as claimed, even when we use estimates of the idle temperature $\hat{T}_{\text{idle}}$ and environmental temperature $\hat{T}_o$, it is from the following equation:

$$\Delta T_R = T_R - (\hat{T}_o + \hat{T}_{\text{idle}}) = T(k) - (\hat{T}_o + \hat{T}_{\text{idle}}) = \Delta T_{\text{fb}}(k)$$

that we have $T_R = T(k)$, that is, the processor's temperature converges to the temperature set point.

It is noted that due to the minimum task rate constraints, there exists a lower bound for the feasible utilization, which in turn results in a lower bound for the feasible temperature. The lower bounds for the utilization and temperature are related to the rate constraints, the actual execution times, and the actual power consumption. TCUB can achieve satisfactory thermal and real-time performance only if both the given temperature set-point and the utilization bound are feasible under the task rate constraints.

V. Evaluation

The simulation environment consists of two components: an event driven simulator implemented in C++ and a Simulink© model implemented in MATLAB (R2008a). The simulator simulates a single processor real-time system controlled by TCUB and implements a utilization monitor, a rate actuator and a utilization controller. The Simulink© component implements the thermal controller and the model of thermal dynamics of the processor. The simulator and the Simulink© component communicate with each other through a TCP connection.

In our simulation the task set running on the processor consists of 10 periodic soft real-time tasks. The Rate Monotonic (RM) scheduling algorithm [14] is employed to schedule all these tasks. Initially, the period of each task T_i is randomly generated in the range $[100ms, 200ms]$. Based on the initial tasks rate, the execution time of tasks are chosen in the way that each task has nearly equal utilization and the collective utilization of all tasks is close to the schedulable utilization bound. The minimum rate of one task equals its execution time while the maximum rate equals 10 times of initial tasks rate. The deadline of each task equals its period.

The processor simulated in our work is a $2.6GHz$ Pentium 4 (P4) processor with $130nm$ Northwood core. All thermal related parameters, except thermal capacitance, shown in Table I are based on Intel technical specification [24]. The thermal capacitance is acquired by simulating P4 on Hotspot [25], an architecture level simulator.

TABLE I
Power and Thermal Parameters

Parameter	Notation	Value
Ambient temperature	T_0	$45°C$
Max case temperature	T_c	$75°C$
Estimated Active power	P_a	$51.9W$
Idle power*	P_i	$13.3W$
Thermal Capacitance	C_{th}	$295.7J/K$
Thermal Resistance	R_{th}	$0.467K/W$
Thermal Fault Resistance	R'_{th}	$2R_{\text{th}}$

* Enhanced Halt Mode is available [26]

In the following simulations, we choose $70°C$ as the set-point of the processor's temperature. The set-point is lower than the maximum case temperature to avoid surpassing the maximum case temperature during dynamic regulation. The thermal fault resistance, R'_{th}, is based on the estimated thermal RC model presented in [2].

Table II shows the controller parameters of TCUB which are calculated using the methods discussed in Section IV.

We compare TCUB against three baseline algorithms[2], OPEN, TC and FC-U. OPEN statically set task rates based on the *estimated* execution times to achieve the schedulable utilization bound (which is higher than the utilization bound U_{max} adopted by TCUB. OPEN represents a static approach commonly used in practice. TC has the same thermal controller as TCUB, but does not include the utilization controller. After the thermal controller outputs the utilization set-point, it sets the task rates based on the *estimated* execution times. FC-U [27] employs the same utilization control algorithm used in TCUB, but does not has the thermal controller to manage temperature. As subsets of TCUB, TC and FC-U allow us to evaluate the effectiveness of the *integrated* control approach of TCUB for both temperature and utilization.

[2]While several thermal-aware real-time scheduling algorithms exist in the literature [4]–[6], [27], they rely on Dynamic Voltage and Frequency Scaling (DVFS) which is not required by TCUB. The only existing feedback control algorithm for thermal control [9] also require on DVFS and hence will not provide a fair comparison with TCUB. We discuss the related work in detail in Section VI.

TABLE II
TCUB Controller Parameters

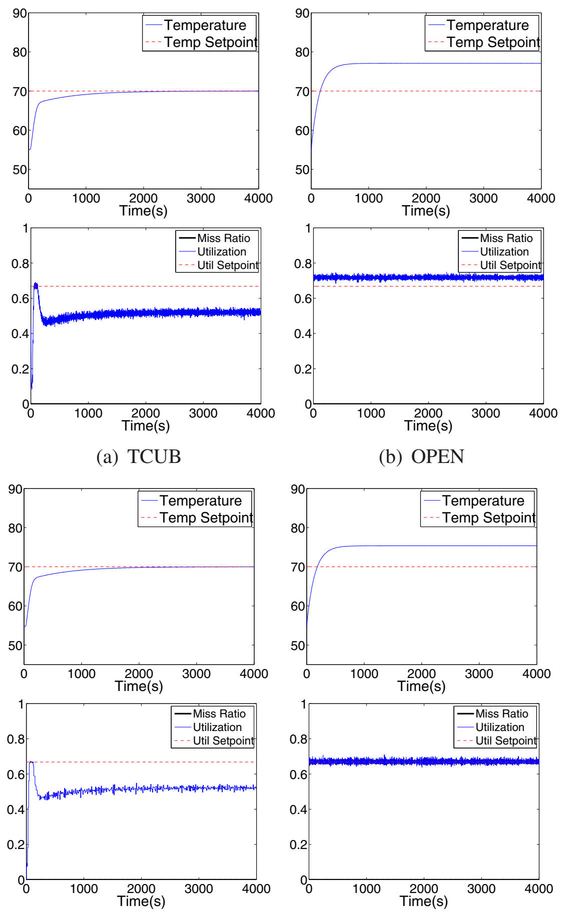
Controllers	Parameters	Value
Thermal Controller	K_p	0.0523
	K_i	0.0523
	ω_i	0.0036
	$k_{p\,\max}$	510
	$R_{\text{th}\,\max}$	0.934
	$U_{\max}$	0.67
	T_R	$70°C$
	T_s	10s
Utilization Controller	K_p	0.37
	T_u	1s

A. Experiment I: Power Deviation

This set of simulations is designed to evaluate TCUB when the processor's active power deviate from the estimate, which represent the phase change of the power in the processor observed in previous empirical studies [1]. We use different *power ratios*, i.e., the ratio between the actual and estimate active power, in different runs. In the first run the power ratio is 2, i.e., the actual active power is twice the estimate; in the second run, the power ratio is 0.5, i.e., the actual power is half of the estimate. The task execution times are the same as their estimate in this set of experiments.

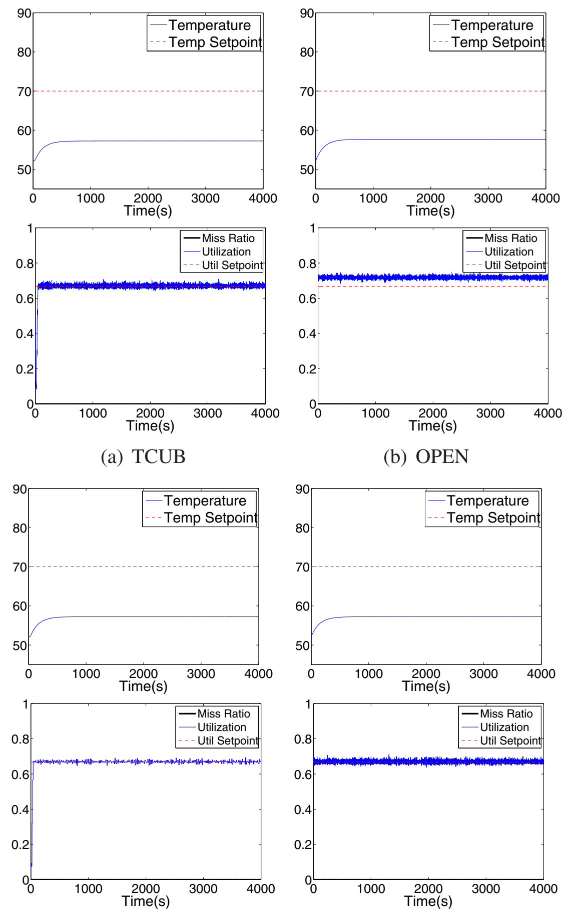
Fig. 4 show the simulation resutls when the power ratio is 2. In Fig. 4(a), the temperature under TCUB converges to the temperature set-point $70°C$, while its utilization is still below the utilization bound. Note that TCUB forces the CPU utilization *lower* than its utilization bound, which is necessary to maintain the temperature set-point in case of high active processor power. In contrast, although FC-U (seeing Fig. 4(c)) maintains the utilization bound it *violates* the temperature set-point. OPEN behaves similarly to FC-U except it achieves slightly higher utilization and temperature because the task rates are initiated as the schedulable utilization bound which is higher than the utilization bound adopted by FC-U. TC performs similarly to TCUB. This is because the execution times are the same as their estimate in this experiment, and hence utilization control is not necessary. There is no deadline miss under all algorithms in this experiment.

Fig. 5 illustrate the simulation results when power ratio is 0.5. TCUB undershoots the temperature set-point while reaching the utilization bound in this experiment. Due to the low processor power, the utilization bound constraint is activated before the temperature reaches the set-point. As a result, TCUB stops increasing the utilization to enforce the utilization bound. TC behaves similarly to TCUB because the



(a) TCUB (b) OPEN (c) TC (d) FC-U

Fig. 4. Performance Comparison (Power Ratio = 2)



(a) TCUB (b) OPEN (c) TC (d) FC-U

Fig. 5. Performance Comparison (Power Ratio = 0.5)

task execution times conform to the estimation. FC-U enforces the utilization bound, which results in a temperature lower than the set-point. OPEN behaves similarly to FC-U.

In summary, this set of experiments demonstrate our thermal controller can effectively handle uncertainties in power consumption.

B. Experiment II: Execution Time Variation

This set of experiments is designed to evaluate TCUB under uncertainties in task execution times. We use *execution-time factor (etf)* to denote the ratio between the actual and the estimated execution times. For example, when etf is 2, the actual execution time is twice the estimate. We simulate the case of $etf = 2$ in this set of experiments. Note that for these experiments the power ratio is 1 , i.e., the processor's active powers is the same as the estimate.

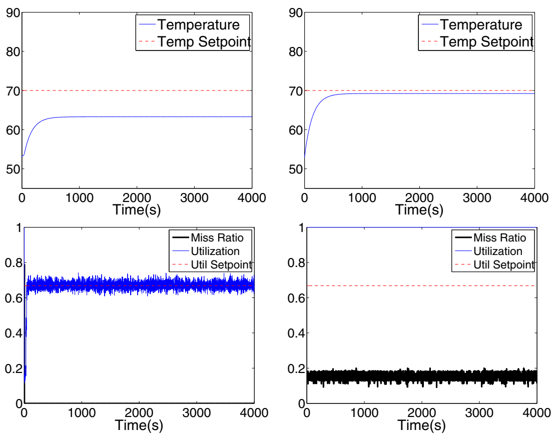
The results are shown in Fig. 6. For TCUB the temperature is below the set-point, while the utilization reaches the schedulable bound. Since power ratio is 1 in this experiment, the utilization bound constraint is activated before the processor's temperature reaches the set-point. TCUB successfully enforces the utilization bound even when the actual execution times exceed their estimate by 100%. No deadline miss is observed under TCUB. This result demonstrates that TCUB effectively handles uncertainties in task execution times through the utilization controller. Similarly, FC-U enforces the utilization bound. In contrast, TC causes the utilization to reach 1.0 as well as a significant number of deadline misses since it adjusts task rates based on their *estimated* execution times. OPEN also resulted in deadline misses due to the deviation of task execution times from the estimate.

Collectively, the first two sets of experiments demonstrate TCUB is the only algorithm in our study that can consistently maintain both acceptable temperature and soft real-time performance under uncertainties of power consumption and task execution times.

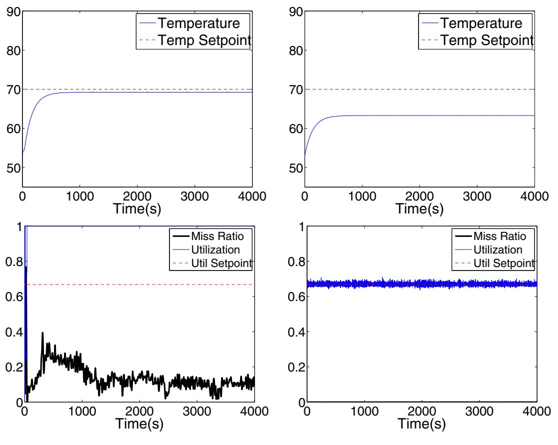
C. Experiment III: Robustness of TCUB

This set of experiments is designed to stress-test the robustness of TCUB under uncertainties of *both* execution times and power consumption. For all the experiments we plot the average temperature and utilization over the last 300 sampling period to exclude the transient effect response in the beginning of the experiments.

Fig. 7 demonstrates the robustness of TCUB when both the execution time factor and the power ratio vary in a wide region. The area filled by circles, i.e., *empirical* area, represents the results in which TCUB satisfies the criteria of average temperature ($\leq 70.7°C$) and average utilization (≤ 0.677), which are 1.01 times of temperature and utilization set-point



(a) TCUB (b) OPEN



(c) TC (d) FC-U

Fig. 6. Performance Comparison ($etf = 2$)

respectively. The *theoretical* bound for the execution time factor is the analytical maximum execution time factor below which the utilization controller can maintain stability based on the analysis presented in [28]. The *theoretical* bound of the power ratio is the maximum power ratio below which our thermal controller can maintain stability based on Theorem 2. The *feasible* bound is determined based the minimum task rates of our workload as discussed in Section IV-C. The area surrounded by the theoretical and the feasible bound is the area within which our system is stable based on our analysis. we call it *analytical* area. As shown in Fig. 7, the empirical area matches well with the analytical area. These results demonstrate that TCUB can maintain desirable temperature and utilization under considerable uncertainties in terms of both power consumption and execution times. Furthermore, the close match between the analytical and empirical area demonstrates the effectiveness of our control model and analysis.

D. Experiment IV: Thermal Fault

This set of experiments is designed to examine the capability of TCUB to deal with thermal faults based on the empirical model presented in [2]. We simulate the case fan failure by doubling the thermal resistance, R_{th}, of the processor. As

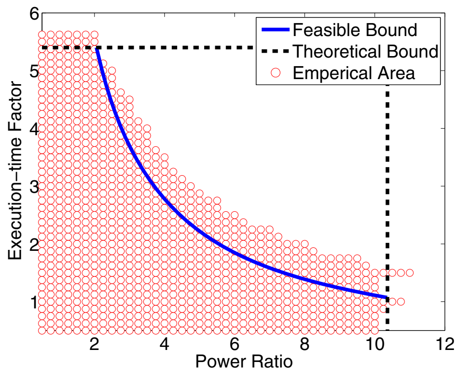
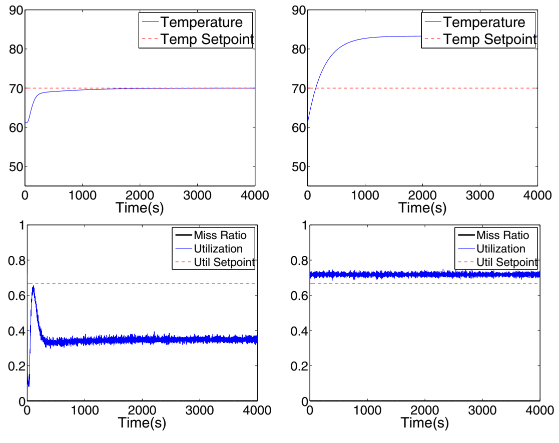
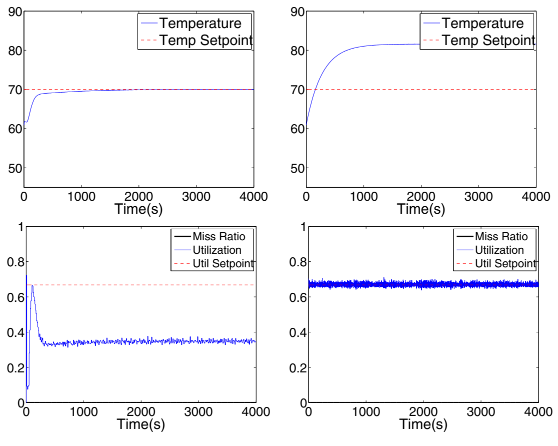


Fig. 7. TCUB Performance with Varying Power Ratio and ETF

shown in Fig. 8, under TCUB the temperature converges to $70°C$ while the utilization remain considerably lower than the utilization bound. Since the thermal resistance doubles in this case, the processor generates more heat at the same utilization and TCUB enforces the temperature set-point by enforcing a low level utilization. TC performs similarly to TCUB as the utilization bound is not activated when it converges to the temperature set-point. In contrast, both FC-U and OPEN significantly overshoot the temperature set point.



(a) TCUB (b) OPEN

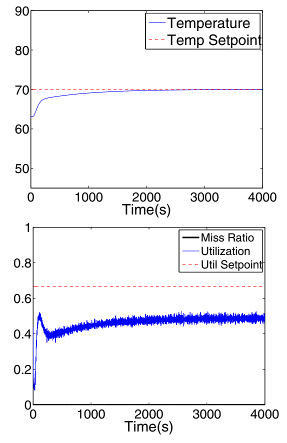


(c) TC (d) FC-U

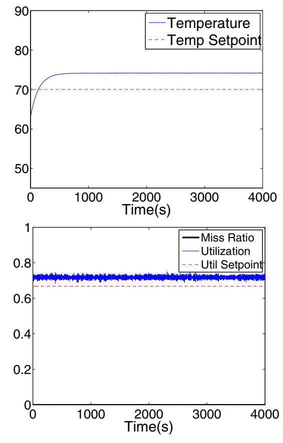
Fig. 8. Performance Comparison with Thermal Fault

E. Experiment V: Ambient Temperature Variation

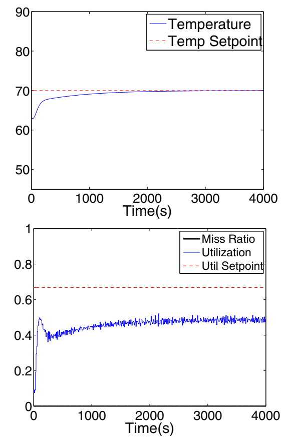
This set of experiments is designed to evaluate TCUB when the ambient temperature is higher than the default setting by $10°C$. The power ratio and etf is fixed at 1.0. As shown in



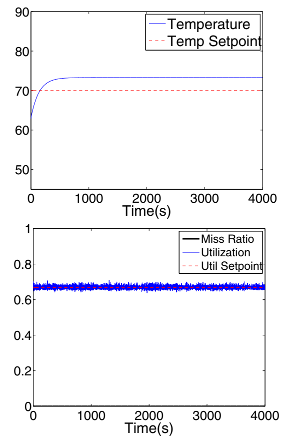
(a) TCUB



(b) OPEN



(c) TC



(d) FC-U

Fig. 9. Performance Comparison with Different Ambient Temperature

Fig. 9(a), TCUB ensures the temperature set-point while the utilization is below the set-point. To compensate the increase of ambient temperature change, TCUB maintains a low level of utilization to reduce the amount of heat generated by the processor. TC behaves similarly to TCUB. In contrast, both FC-U and OPEN exceed the temperature set-point with high utilization.

VI. Related Work

Thermal-aware real-time scheduling has received attention recently. Existing single-processor scheduling algorithms [4]–[8] exploit DVFS to enforce temperature bounds while meeting task deadlines. Thermal-aware tasks relocation and scheduling algorithms have also been proposed for multi-processor or multi-cores systems [27], [29]. Despite significant research on thermal-aware real-time scheduling, existing algorithms rely on accurate knowledge about the system characteristics such as task execution times, power consumption, and ambient temperature, which can vary at run time for real-time systems operating in unpredictable environments. In sharp contrast, thanks for its robust feedback control approach TCUB is specifically designed to handle a broad range of uncertainties dynamically. In addition, TCUB does not rely on DVFS to control processor temperature, which makes it a practical solution even for embedded processors that do not support DVFS.

The most related to our work is [9] which proposed a model-predictive control approach for thermal and utilization control in distributed real-time systems. While sharing similar goals as TCUB, there are several major differences between that work and TCUB. First, the algorithm proposed in [9] uses different actuators to control temperature (DVFS) and utilization (task rate adaptation). Instead, TCUB uses only task rate adaptation to control both temperature and utilization. This not only makes TCUB a more general solution, but also poses unique challenges as temperature and utilization control are closely coupled in our system due to the shared actuator. Second, our control design is fundamentally different from the model predictive control approach taken in [9]. We significantly simplified the control problem by explicitly enforcing the utilization bound by including it into an integrator-anti-wind-up thermal control strategy. Our novel control design result in a simple and efficient nested control algorithm with $\mathcal{O}(1)$ run-time overhead. In contrast, the model predictive controller [9] rely on a least-squares estimator with polynomial complexity to the product of the number of tasks and the control and prediction horizons. The simplicity and efficiency of TCUB make it a practical solution even for resource-limited embedded processors. Finally, our simple control approach allows rigorous robustness analysis. Since our robustness analysis is based on the necessary and sufficient conditions required of the Nyquist stability criteria, we prove and demonstrate how our controller can respond quickly while operating under a wide range of system uncertainties. In contrast, the small-gain conditions [30] required to satisfy robustness criteria of the proposed model -predictive- controller presented in [9] tend to be conservative and computationally intensive to verify [31]. Loosening these model uncertainty constraints for model-predictive controllers is a daunting task as noted in [32] and currently being addressed in [33]–[35].

A multitude of feedback real-time scheduling and utilization control algorithms have been proposed in recent years, [36]–[42], but they are not cognizant of processor temperature. In contrast, TCUB is designed to control *both* the real-time performance and the processor temperature. While TCUB incorporates a utilization controller, the key contribution of this work is the nested control architecture and the novel thermal controller that can handle the utilization bound constraint needed to enforce desired soft real-time performance.

VII. Conclusion

Many embedded systems face the critical challenge of managing both the processor temperature and software real-time performance in unpredictable environments. This paper presents TCUB, a control-theoretic algorithm for managing both the processor temperature and real-time performance. Rigorously modeled and designed based on control theory, TCUB can avoid processor overheating and maintain soft real-time performance. A salient feature of TCUB lies in its capability to handle different types of uncertainties in terms of (1) processor power consumption, (2) task execution times, (3) ambient temperature, and (4) unexpected thermal faults. The robustness of TCUB makes it particularly suitable for real-time embedded systems that must deal with highly unpredictable environments. Moreover, TCUB features a nested feedback control structure consisting of (1) a low-rate thermal controller dealing with the slower thermal dynamics, and (2) a high-rate utilization controller handling the faster CPU utilization dynamics caused by uncertainties in task execution times. The nested control scheme is modular, efficient, and practical for embedded systems with tight resource constraints. The advantages of TCUB have been demonstrated through extensive simulations under a broad range of system and environmental conditions.

Acknowledgment

The research of Yong Fu, Yingming Chen and Chenyang Lu is supported in part by NSF CAREER Award CNS-0448554. The research of Nicholas Kottenstette and Xenofon Koutsoukos is supported in part by NSF Award NSF-CCF-0820088. The research of Hongan Wang is supported in part by Chinese National Programs for High Technology Research 2009AA01Z337 and Development and Key Project of Chinese National Programs for Fundamental Research and Development 2009CB320804.

References

[1] C. Isci and M. Martonosi, "Runtime power monitoring in high-end processors: Methodology and empirical data," in *MICRO*, 2003.

[2] A. P. Ferreira, D. Mosse, and J. C. Oh, "Thermal faults modeling using a RC model with an application to Web farms," in *ECRTS*, 2007.

[3] C. Lu, J. A. Stankovic, G. Tao, and S. H. Son, "Feedback control real-time scheduling: Framework, modeling, and algorithms," *Real-time Systems*, vol. 23, 2002.

[4] S. Wang and R. Bettati, "Reactive speed control in temperature-constrained real-time systems," in *ECRTS*, 2006.

[5] ——, "Delay analysis in temperature-constrained hard real-time systems with general task arrivals," in *RTSS*, 2006, pp. 323–334.

[6] W. Hung, Y. Xie, N. ViJáykrishnan, M. Kandemir, and M. J. Irwin, "Thermal-aware task allocation and scheduling for embedded systems," in *DATE*, 2005, pp. 898–899.

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

[8] L. Yuan, S. Leventhal, and G. Qu, "Temperature-aware leakage minimization techniques for real-time systems," in *ICCAD*, Nov. 2006.

[9] X. Fu, X. Wang, and E. Puster, "Dynamic thermal and timeliness guarantees for distributed real-time embedded systems," in *RTCSA*, 2009.

[10] A. Cervin, J. Eker, B. Bernhardsson, and K.-E. Arzen, "Feedback feedforward scheduling of control tasks," *Real-Time System*, vol. 23, no. 1-2, pp. 25–53, 2002.

[11] S. Brandt and G. J. Nutt, "Flexible soft real-time processing in middleware," *Real-Time System*, vol. 22, no. 1-2, 2002.

[12] V. Tiwari, S. Malik, and A. Wolfe, "Power analysis of embedded software: A first step towards software power minimization," *IEEE Transactions on VLSI Systems*, vol. 2, pp. 437–445, 1994.

[13] F. Bellosa, S. Kellner, M. Waitz, and A. Weissel, "Event-driven energy accounting of dynamic thermal management," in *COLP*, 2003.

[14] C. Liu and J. Layland, "Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment," *JACM*, vol. 20, no. 1, pp. 46–61, 1973.

[15] T. Heath, A. P. Centeno, P. George, L. Ramos, Y. Jaluria, and R. Bianchini, "Mercury and freon: temperature emulation and management for server systems," in *ASPLOS*, 2006.

[16] G. F. Franklin, J. D. Powell, and M. Workman, *Digital Control of Dynamic Systems*. Addison Wesley Longman, Inc., 1998.

[17] N. Kottenstette, J. Hall, X. Koutsoukos, P. Antsaklis, and J. Sztipanovits, "Digital control of multiple discrete passive plants over networks," *International Journal of Systems, Control and Communications (IJSCC)*, no. Special Issue on Progress in Networked Control Systems, 2009, to Appear.

[18] N. Kottenstette, H. LeBlanc, E. Eyisi, and X. Koutsoukos, "Multi-rate networked control of conic systems," Institute for Software Integrated Systems, Vanderbilt University, Nashville, TN, Tech. Rep., 09/2009 2009.

[19] G. F. Franklin, J. D. Powell, and A. Emami-Naeini, *Feedback Control of Dynamic Systems*, 5th ed. Prentice-Hall, 2006.

[20] A. Oppenheim, A. Willsky, and S. Nawab, *Signals and systems*. Prentice hall Upper Saddle River, NJ, 1997.

[21] Y. Fu, N. Kottenstette, Y. Chen, C. Lu, X. Koutsoukos, and H. Wang, "Feedback thermal control for real-time systems," Department of Computer Science and Engineering, Wahsington University in St. Louis, Tech. Rep. 2009-17, 2009.

[22] G. Herrmann, M. Turner, and I. Postlethwaite, "Discrete-time and sampled-data anti-windup synthesis: stability and performance," *International Journal of Systems Science*, vol. 37, no. 2, pp. 91–113, 2006.

[23] G. Grimm, A. Teel, and L. Zaccarian, "The l2 anti-windup problem for discrete-time linear systems: Definition and solutions," *Systems & Control Letters*, vol. 57, no. 4, pp. 356–364, 2008.

[24] Intel Corp., "Intel Pentium 4 processor in the 423-pin package thermal design guidelines," Intel, Tech. Rep., 2000.

[25] W. Huang, M. R. Stan, K. Skadron, K. Sankaranarayanan, S. Ghosh, and S. Velusam, "Compact thermal modeling for temperature-aware design," in *DAC*, 2004.

[26] Tom's Hardware, "3.8 GHz P4-570 and e0 stepping to end Intel's performance crisis," Tom's Hardware, Tech. Rep., 2004. [Online]. Available: http://www.tomshardware.com/reviews/3,920-21.html

[27] T. Chantem, R. P. Dick, and X. S. Hu, "Temperature-aware scheduling and assignment for hard real-time applications on mpsocs," in *DATE*, 2008.

[28] X. Wang, Y. Chen, C. Lu, and X. Koutsoukos, "Towards controllable distributed real-time systems with feasible utilization control," *IEEE Transactions on Computers*, vol. 58, no. 8, pp. 1095–1110, 2009.

[29] N. Fisher, J.-J. Chen, , S. Wang, and L. Thiele, "Thermal-aware global real-time scheduling on multicore systems," in *RTAS*, 2009.

[30] E. Zafiriou, "Robust model predictive control of processes with hard constraints." *Comp. Chem. Eng.*, vol. 14, no. 4, pp. 359–371, 1990.

[31] A. Zheng, "Robust stability analysis of constrained model predictive control," *Journal of Process Control*, vol. 9, no. 4, pp. 271–278, 1999.

[32] D. Mayne, J. Rawlings, C. Rao, and P. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, pp. 789–814, 2000.

[33] G. Grimm, M. Messina, S. Tuna, and A. Teel, "Nominally robust model predictive control with state constraints," *IEEE Transactions on Automatic Control*, vol. 52, no. 10, pp. 1856–1870, 2007.

[34] C. Løvaas, M. Seron, and G. Goodwin, "Robust output-feedback model predictive control for systems with unstructured uncertainty," *Automatica*, vol. 44, no. 8, pp. 1933–1943, 2008.

[35] M. Lazar, D. Muñoz de la Peña, W. Heemels, and T. Alamo, "On input-to-state stability of min–max nonlinear model predictive control," *Systems & Control Letters*, vol. 57, no. 1, pp. 39–48, 2008.

[36] S. Wang and R. Bettati, "Reactive speed control in temperature-constrained real-time systems," *Real-Time Systems*, vol. 39, no. 1-3, pp. 73–95, 2008.

[37] J. A. Stankovic, T. He, T. Abdelzaher, M. Marley, G. Tao, S. Son, and C. Lu, "Feedback control scheduling in distributed real-time systems," in *RTSS*, Dec. 2002.

[38] R. M. Santos, G. Lipari, and E. Bini, "Efficient on-line schedulability test for feedback scheduling of soft real-time tasks under fixed-priority," in *RTAS*, 2008.

[39] K.-D. Kang, J. Oh, and S. H. Son, "Chronos: Feedback control of a real database system performance," in *RTSS*, 2007.

[40] Y. Zhu and F. Mueller, "DVSleak: combining leakage reduction and voltage scaling in feedback EDF scheduling," in *LCTES*, 2007.

[41] M. Amirijoo, J. Hansson, S. Gunnarsson, and S. H. Son, "Quantifying and suppressing the measurement disturbance in feedback controlled real-time systems," *Real-Time Systems*, vol. 40, no. 1, pp. 44–76, 2008.

[42] A. Block, B. Brandenburg, J. Anderson, and S. Quint, "Adaptive multiprocessor real-time scheduling with feedback control," in *ECRTS*, Jul. 2009.