SyRaFa: Synchronous Rate and Frequency Adjustment for Utilization Control in Distributed Real-Time Embedded Systems

Xi Chen, Xiao-Wen Chang, and Xue Liu

Abstract—To efficiently utilize the computing resources and provide good Quality of Service (QoS) to the end-to-end tasks in the distributed real-time systems, we can enforce the utilization bounds on multiple processors. The utilization control is challenging especially when the workload in the system is unpredictable. To handle the workload uncertainties, current research favors feedback control techniques, and recent work combines the task rate adaptation and processor frequency scaling in an asynchronous way for CPU utilization control, where task rates and the processor frequencies are tuned asynchronously in two decoupled control loops for control convenience. Since the two manipulated variables, task rates and processor frequencies, contribute to the CPU utilizations together with strong coupling, adjusting them asynchronously may degrade the utilization control performance. In this paper, we provide a novel scheme to make Synchronous Rate and Frequency Adjustment to enforce the utilization setpoint, referred to as SyRaFa scheme. SyRaFa can handle the workload uncertainties by identifying the system model online and can simultaneously adjust the manipulated variables by solving an optimization problem in each sampling period. Extensive evaluation results demonstrate SyRaFa outperforms the existing schemes especially under severe workload uncertainties.

Index Terms—Distributed systems, embedded systems, real-time systems, end-to-end task, quality of service, optimization.

1 Introduction

Distributed real-time systems, such as wireless network, networked embedded systems and Supervisory Control and Data Acquisition (SCADA) systems, are gaining increasingly importance and growing more complex with abundant computing devices being connected together. These systems often work in an open environment with dynamic workload. Efficiently using the computing resources in these systems is of vital significance in order to provide quality of service (QoS) guarantees and at the same time reduce the energy consumption. CPU utilization control is a good solution to reach the goal. In the distributed real-time systems holding the tasks with soft end-to-end deadlines, keeping the CPU utilization of each processor close to its schedulable utilization bound can usually provide small miss-deadline ratio to the real-time tasks. Moreover, CPU utilization control can lower down the energy consumption in the processors by avoiding over-provision of the computing resources to the real-time tasks. It is challenging to conduct CPU utilization control in distributed real-time systems with workload variations and uncertainties. To handle the workload uncertainties, most recent work favors feedback control techniques for CPU utilization control [1]–[6], since the feedback control techniques can tolerate the system modeling error caused by workload variations and can leverage the measured processor utilizations as continuous feedback to properly enforce the utilization setpoint of each processor.

- *Xi Chen, Xiao-Wen Chang and Xue Liu are with the School of Computer Science, McGill University, Montreal, Quebec, Canada.*
  E-mail: xi.chen7@mail.mcgill.ca, chang,xueliu@cs.mcgill.ca

The percentage of CPU time utilized by a task in a processor is the product of the task execution time and the task rate. Given a set of tasks located on each processor, the CPU utilization of this processor can be affected by adjusting the execution times and the task rates. In modern distributed real-time systems where the processors support dynamic voltage and frequency scaling (DVFS) techniques, we can adjust the execution times of the tasks on each processor by scaling the processor frequency [4]. Moreover, the task rates can be tuned within the specified ranges by the rate modulator on the processors. Therefore, processor frequency scaling can be combined with task rate adaptation to realize CPU utilization control.

Typical work on CPU utilization control by using processor frequency scaling and task rate adaptation was proposed in [4]. The scheme in [4] leverages feedback control techniques to adjust the two manipulated variables, processor frequencies and task rates, to enforce the utilization setpoints, which outperforms previous utilization control schemes that depend on task rate adaptation exclusively such as EUCON [1]. Since simultaneously adjusting the two manipulated variables will result in a nonlinear control model, [4] decomposes the system model into two linear control loops for CPU utilization control; one control loop uses task rate adaptation and the other uses processor frequency scaling. The decoupled control loops are controlled in their respective sampling periods without interactions. Although the feedback control technique can handle the workload uncertainties to certain extent, the decoupled and asynchronous adjustments to the manipulated variables is not an ideal way to conduct CPU utilization control. This is because task rates and processor frequencies contribute to the CPU utilization with strong mutual influence. Ignoring the coupling between the two manipulated variables to cater the feedback control techniques may degrade the utilization control performance.

In this paper, we propose a Synchronous Rate and Frequency Adjustment scheme, referred to as SyRaFa, for utilization control. To preserve the coupling between the two manipulated variables, we simultaneously adjust the task rates and processor frequencies by solving a bilinear mixed-discrete least squares (LS) optimization problem online instead of using the feedback control techniques. Since solving the LS problem calls for a highly accurate system model, we use the measurement to the CPU utilization of the processors as feedback information to estimate the parameters in the system model online. Specifically, we develop a hybrid strategy that adopts a workload change point detection approach and a Recursive Least Squares (RLS) approach to predict the load-variation matrix in the utilization model adaptively.

The main contribution of this paper can be summarized as follows:

- We propose SyRaFa, a novel solution based on a bilinear mixed-discrete LS problem method for optimal processor frequencies and task rates adjustment to enforce desired CPU utilization. In contrast to the typical asynchronous Rate and Frequency (R&F) control scheme [4], our scheme takes advantage of the interaction between task rates and processor frequencies instead of decoupling their relation [4] to control the processor utilization.
- We propose the G-estimator for workload estimation. This RLS-based estimation approach with change point detection can handle the large workload uncertainties and fluctuations, and it can filter out the white noises in the measurement of the processor utilizations.

Experimental results show that SyRaFa outperforms the asynchronous R&F control scheme especially when large workload uncertainties and fluctuations exist. Moreover, the computation cost of SyRaFa is small for online implementation.

The rest of this paper is organized as follows. Related work is discussed in Section 2. We state CPU utilization problem in distributed real-time systems and formulate the model for utilization control in Section 3. The SyRaFa scheme is given in Section 4. We introduce the G-estimator for online estimation to the load-variation matrix in Section 5. We discuss the utilization control for synchronous adjustment to the manipulated variables in Section 6. Section 7 demonstrates simulation results of SyRaFa in comparison with the asynchronous R&F control scheme. Section 8 summarizes the main results and concludes the paper.

2 Related Work

To ensure the End-to-End deadlines of the hard real-time tasks in the distributed systems, typical scheduling algorithms like end-to-end scheduling [7] and distributed scheduling were provided [8], [9]. However, these algorithms are open-loop approaches designed offline based on the worst-case execution time (WCET) of all the tasks. With increasing distributed systems working in the open and unpredictable environment, the open-loop approaches based on worst-case estimation to task execution time may result in the underutilization of the processors and waste energy. Moreover, many of these distributed real-times systems contain the tasks with soft end-to-end deadlines, missing deadlines at a small ratio can also provide satisfactory QoS to the real-time tasks. Therefore, it is possible to trade off the end-to-end deadline guarantees for the efficient usage of the system energy and computing resources.

Recently, control-theoretic approaches have been proposed to handle the workload uncertainties and provide desired QoS in general for computing systems, and for distributed systems holding soft real-time tasks in particular. Surveys on feedback control for resource management in real-time computing systems are presented in [10] and [11]. Many projects and applications on feedback scheduling are implemented. In particular, Goel et al. [12] proposed a feedback-based scheduling approach to meet real-time requirement. Stankovic et al. [13] applied feedback control to schedule real-time systems. Wang et al. [14] proposed a middleware that adopts feedback control techniques for CPU utilization and miss deadline ratio control. Fu et al. [6] used a distributed utilization feedback controller to handle system dynamics caused by load balancing for large-scale server clusters.

Most previous work on CPU utilization control in distributed real-time systems is based on task rate adaptation exclusively. To enforce desired utilization by adjusting the task rates, various control methods are leveraged. Wang et al. used Model Predictive Control (MPC) to handle execution time variations within a specified range in [1], [3] and [4]. Yao et al. [5] developed an adaptive control method to handle large workload uncertainties. However, all these algorithms assume that task rates can be continuously tuned. To tackle the discrete task rates, hybrid control theory [15] and optimization strategies [16] were proposed, but they are based on the priori knowledge of the tasks' WCET and are not applicable to the unpredictable environment. Using task rate adaption exclusively for utilization control may degrade control performance due to reasons such as discrete rate truncation or task rate saturation.

DVFS has been leveraged as an effective technique for energy efficient scheduling. Many of the energy-aware scheduling algorithms are proposed [17]–[22]. However, most of them are open-looped algorithms that rely on priori knowledge of WCETs, and they focus on minimizing energy or temperature and do not use DVFS as a technique for utilization control.

In contrast to the above utilization control methods and DVFS-based scheduling algorithms, Wang et al. [4] proposed a power-aware utilization control scheme that uses DVFS for utilization control. In [4], processor frequency scaling is used as a compensation for task

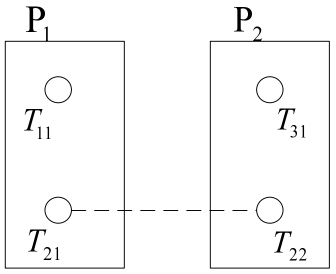


Fig. 1: Distributed Real-time System Model

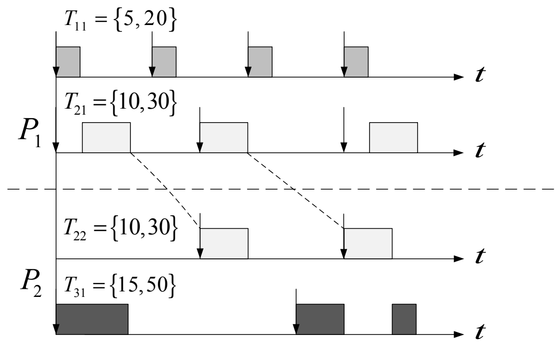


Fig. 2: Scheduling of Subtasks

rate adaption. However, it separates the two strongly-coupled variables, processor frequencies and task rates, into two decoupled linear control models, which may degrade the overall system performance especially when large workload variations exist. In this paper, we propose SyRaFa, which adopts the task rate adaptation and processor frequency scaling synchronously for CPU utilization control. SyRaFa preserves the nonlinear system model and leverages the coupling of the manipulated variables to conduct better utilization control.

3 Utilization Problem in Distributed Real-time Systems

In this section, we first state the CPU utilization problem. Based on the CPU utilization problem, we provide the model for utilization control.

3.1 CPU Utilization Problem in Distributed Real-time Systems

We consider the same task model as [1] for forming the CPU utilization problem in distributed real-time systems.

A distributed real-time system consists of m independent and preemptable end-to-end tasks $\{T_i|1 \leqslant i \leqslant m\}$ executed on n processors $\{P_q|1 \leqslant q \leqslant n\}$. Each task T_i has a chain of subtasks $\{T_{ij}|1 \leqslant i \leqslant m, 1 \leqslant j \leqslant t_i\}$ that could be located on multiple processors, where t_i is the number of subtasks held by task T_i. A typical distributed real-time system with 2 processors and 2 end-to-end tasks (with totally 4 subtasks) is shown in Fig. 1.

Each task T_i is a periodic task, i.e., instances of its first subtask T_{i1} are released periodically. The subtasks of the same task T_i should be executed sequentially, i.e. subtask $T_{i,j+1}$ cannot be released until its predecessor T_{ij} completes. We assume that the non-greedy synchronization protocol like Phase Modification (PM) protocol [7] is applied to enforce the precedence constraints on the subtasks. The PM protocol ensures the instances of every subtask T_{ij} in T_i are released periodically with the same period and the proof is in [7]. Each subtask T_{ij} can be described by a tuple $\{c_{ij}, p_i\}$, where c_{ij} is the estimated execution time of T_{ij} and p_i is the task period. The relative deadline of the subtask equals to p_i. The relative end-to-end deadline of the task T_i is the sum of all the relative deadlines of its subtasks. Subtasks located on the same processor are scheduled by the fixed-priority scheduling algorithm, and we use rate monotonic scheduling (RMS) [23] in this paper. Consider the example in Fig. 1, assume $\{c_{ij}, p_i\}$ for each subtask T_{ij} are $T_{11} = \{5, 20\}$, $T_{21} = \{10, 30\}$, $T_{22} = \{10, 30\}$, $T_{31} = \{15, 50\}$. Since T_{21} and T_{22} are subtasks of T_2, we apply PM protocol to set the releasing phase of T_{22} as its period, hence the first instance of T_{22} is released 30 time units after the origin of the runtime. Under PM protocol, the instances of T_{21} and T_{22} are released periodically with the same period. In this case, as long as we can ensure the schedulability of T_{21} and T_{22}, each instance of T_{22} can never be released before the fulfillment of the corresponding instance of its predecessor T_{21}. Both P_1 and P_2 use RMS to schedule the periodic subtasks. On P_1, T_{11} is given higher priority than T_{21} since it has shorter task period. Similarly, T_{22} is given higher priority than T_{31} on P_2. The scheduling of the subtasks on P_1 and P_2 are given in Fig. 2. According to [23], the subtasks can meet their deadlines if the utilization of each processor stays under a schedulable utilization bound. The local schedulability of the subtasks helps to improve the schedulability of the end-to-end tasks, although cannot completely guarantee.

Modern processor supports CPU frequency scaling technique like DVFS or clock modulation, and the CPU frequency can be adjusted within a continuous range [4]. We use f_q to denote a normalized CPU frequency, which is a ratio of the CPU frequency of processor P_q to its highest value, and f_q can be adjusted within the range $[f_{min}, 1]$. For each subtask T_{ij} that is executed on P_q, the execution time of T_{ij} is inversely proportional to f_q and can be estimated as $\frac{c_{ij}}{f_q}$, where c_{ij} is the estimated execution time of T_{ij} when P_q runs at its highest CPU frequency, namely $f_q = 1$. All the subtasks T_{ij} from the same task T_i share the same task period p_i. We use the term "task rate" to denote the inverse of task period, denoted by r_i, where $r_i \triangleq \frac{1}{p_i}$. The task rate r_i can be adjusted within a predefined set $\mathcal{R}_i \triangleq \{\gamma_{i1}, \gamma_{i2}, \ldots, \gamma_{il}\}$, where l is the number of discrete levels in the set and is assumed to be the same for all the tasks in this paper.

Given S_q as the set of all subtasks T_{ij} executed on processor P_q, the utilization of each processor P_q can be estimated by

$$\hat{u}_q = \frac{1}{f_q} \sum_{T_{ij} \in S_q} c_{ij} r_i. \tag{1}$$

The distributed system may work in open environment with dynamic workload, the processor utilization may deviate from its estimated value from time to time. we define g_q as the actual processor utilization u_q to its estimated value $\hat{u}_q$ (see Eq. (1)), so

$$g_q \triangleq \frac{u_q}{\hat{u}_q}.$$

g_q reflects the workload variation. Therefore, we have:

$$u_q = \frac{g_q}{f_q} \sum_{T_{ij} \in S_q} c_{ij} r_i, \tag{2}$$

Let $u \triangleq [u_1, u_2, ..., u_n]^T$ be the utilization vector of the n processors in the distributed real-time systems, then we have:

$$u = G{*}F{*}E{*}r, \tag{3}$$

where $G \triangleq \mathrm{diag}(g_1, g_2, \ldots, g_n)$ referred to as the load-variation matrix, $F \triangleq \mathrm{diag}(1/f_1, 1/f_2, \ldots, 1/f_n)$ referred to as the frequency inverse matrix, $E \triangleq (e_{ij}) \in \mathbb{R}^{n \times m}$ referred to as a subtask allocation matrix, $e_{ij} \triangleq c_{jk}$ if subtask T_{jk} (the kth subtask of task T_j) is on processor P_i, and $e_{ij} \triangleq 0$ if no subtask of T_j is executed on P_i, $r \triangleq [r_1, r_2, \ldots, r_m]^T$ referred to as the task rate vector. The utilization of each processor can be adjusted by tuning the task rates and the processor frequencies.

In the distributed systems, controlling the CPU utilization of each processor to be close to a utilization setpoint has many profits. On one hand, CPU utilization control can reduce the power consumption by scaling down the processor frequencies when the workload is overestimated, which avoids over-provisioning of the computing resources to the real-time tasks. On the other hand, CPU utilization control can keep the CPU utilization at a setpoint that is slightly lower than the schedulable utilization bound like the RMS utilization bound [23], which can make most of the subtasks on each processor to meet their deadlines. Since each task T_i in the distributed real-time systems has a soft end-to-end relative deadline related to its period which can be divided into deadlines of its subtasks. Providing a small miss-deadline ratio for the subtasks results in a small miss-deadline ratio for the end-to-end tasks.

It worths noticing that we ***do not*** claim the CPU utilization control can ensure schedulability of the end-to-end tasks under the dynamic workload. This is because when the execution times of the subtasks change from time to time, the release interval between consecutive instances from the same subtask may contain jitter due to the sequential relation among the subtasks. CPU utilization control cannot guarantee each task to meet its end-to-end deadline under the workload variations. But we will demonstrate in Section 7 that the CPU utilization control can provide a small miss-deadline ratio to the real-time tasks, which can be tolerated in the system with soft real-time tasks.

3.2 Formulation of CPU Utilization Control

The objective of CPU utilization control is to make the utilization of each processor close to its utilization setpoint with small error at each sampling period during the runtime. Naturally, CPU utilization control problem can be formed as a least squares problem with constraints on the manipulated variables.

The sampling period T_s is selected so that multiple instances of each task may be released during a sampling period [1]. Define the time interval $[(k-1)T_s, kT_s]$ as the kth sampling period, $k = 1, 2, \ldots$. In this interval, the utilization vector is $u(k)$, the frequency inverse matrix is $F(k-1)$, the task rate vector is $r(k-1)$, and the load-variation matrix is $G(k)$. From Eq. (3), the utilization at the kth sampling period becomes:

$$u(k) = G(k){*}F(k-1){*}E{*}r(k-1). \tag{4}$$

As aforementioned, our goal for utilization control is to enforce $u(k)$ to stay at the setpoint $u^s \triangleq [u^s_1, u^s_2, ..., u^s_n]^T$ for $1 \leqslant k \leqslant (\text{Total Runtime})/T_s$. The manipulated variables for utilization control are task rates and processor frequencies. Adjusting the two manipulated variables at each sampling period affects CPU utilization. At the kth sampling period, given the utilization setpoint vector u^s, the task rate set $\mathcal{R}_i$ for each task T_i, and the normalized processor frequency range $[f_{\min}, 1]$ for all the processors, the CPU utilization control objective is to find the task rate vector $r(k)$ and frequency inverse matrix $F(k)$ to minimize the difference between $u(k+1)$ and u^s in the $(k+1)$th sampling period $[kT_s, (k+1)T_s]$. Naturally the 2-norm is used to describe the difference between $u(k+1)$ and u^s, leading to the following LS problem [1]:

$$\begin{aligned} &\min_{F,r} \|u^s - G(k+1){*}F{*}E{*}r\|_2 \\ &\text{subject to} \quad r_i \in \mathcal{R}_i, 1 \leqslant i \leqslant m, \\ &\qquad\qquad\quad f_{\min} \leqslant f_q \leqslant 1, 1 \leqslant q \leqslant n. \end{aligned} \tag{5}$$

For clarity, we summarize the notations used in this paper in Table I in the supplementary document.

4 Overview of the SyRaFa scheme

To enforce the utilization setpoints, we develop a scheme to be referred to as SyRaFa (Synchronous Rate and Frequency Adjustment). Note that $G(k+1)$ in the LS problem (5) is unknown. In order to deal with this problem, we first estimate $G(k+1)$ to get an estimate $\widehat{G}(k+1)$, then we solve the LS problem with $G(k+1)$ replaced by $\widehat{G}(k+1)$, i.e.,

$$\begin{aligned} &\min_{F,r} \|u^s - \widehat{G}(k+1){*}F{*}E{*}r\|_2 \\ &\text{subject to} \quad r_i \in \mathcal{R}_i, 1 \leqslant i \leqslant m, \\ &\qquad\qquad\quad f_{\min} \leqslant f_q \leqslant 1, 1 \leqslant q \leqslant n. \end{aligned} \tag{6}$$

The optimal F and r are denoted by $F(k)$ and $r(k)$, which will be applied to the distributed real-time system to generate $u(k+1)$.

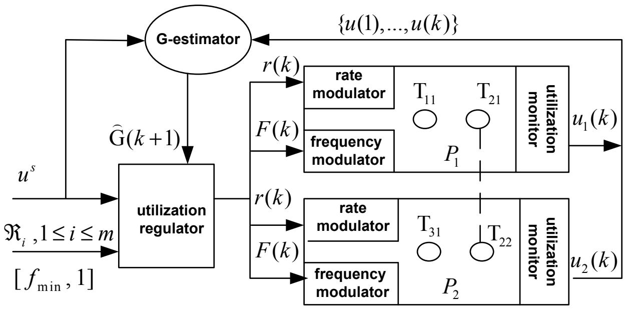


Fig. 3: Overview of SyRaFa

Thus, SyRaFa consists of two parts, the estimator to dynamically estimate workload by predicting the load-variation matrix, and the regulator to synchronously adjust task rates and processor frequencies.

As shown in Fig. 3, SyRaFa is a feedback-based resource allocation scheme that can adjust the task rates and the processor frequencies simultaneously to enforce the utilization setpoints. As the utilization model in Eq. (4) shows, the key to find the desired manipulated variables is to estimate the load-variation matrix $G(k)$ accurately even in volatile environment. We propose the so called G-estimator to predict G online by handling the varying workload. To solve the LS problem in Eq. (6), we use a centralized utilization regulator in the distributed real-time system to compute the optimal manipulated variables. The utilization regulator may locate on a separate processor or share a processor with some subtasks. SyRaFa application (including the G-estimator and utilization regulator) is given the highest priority to be scheduled on the processor that executes it to achieve effective utilization control. Each processor has a $utilization\, monitor$, a $rate\, modulator$ and a $frequency\, modulator$.

For load-variation matrix estimation, the user inputs to G-estimator are u^s, the measured processor utilizations formed in the vector $u(k)$, and the historical utilization measurements $\{u(1), u(2), ..., u(k-1)\}$ that are stored in G-estimator. The output of G-estimator is the estimation to G in the next sampling period (the $(k+1)$th sampling period), $\widehat{G}(k+1)$.

For task rates and processor frequencies computation, the inputs to the utilization regulator are $\widehat{G}(k+1)$ from G-estimator, u^s, $\mathcal{R}_i$ for $1 \leqslant i \leqslant m$ and the normalized processor frequency range $[f_{\min}, 1]$. The outputs are the the manipulated variables $r(k)$ and $F(k)$.

The utilization control procedures are summarized in Section 2.1 of the supplementary document.

5 Online Estimation of Utilization Model

We propose G-estimator to estimate the workload variation matrix $G(k)$ online based on the measured processor utilizations formed in the vector $u(k)$. Since measurement noise may exist, we rewrite the utilization model in Eq. (4) as follows:

$$u(k) = G(k)*F(k-1)*E*r(k-1) + v(k), \qquad (7)$$

where $v(k) \in \mathbb{R}^n$ is the noise vector whose entries are independent and identically distributed n-dimensional random variables with zero means. The working principle of G-estimator is summarized as below. When workload changes abruptly at some sampling period, we detect the workload change and estimate G based on the utilization measurement in this sampling period. When the workload stays unchanged in several consecutive sampling periods, we use recursive least squares (RLS), a widely adopted approach for online model estimation, to estimate G by leveraging the utilization measurements in these sampling periods.

5.1 G-estimation Based on Change Point Detection

At the kth sampling period, the true value of the workload variation is $G(k)$ and the measured utilization $u(k)$ is generated by applying $F(k-1)$ and $r(k-1)$ to the system, hence from Eq. (7) we have

$$u(k) \approx G(k)*F(k-1)*E*r(k-1). \qquad (8)$$

Define

$$d(k) \triangleq F(k-1)*E*r(k-1) \in \mathbb{R}^n, \qquad (9)$$

and let the ith entry of $d(k)$ for $i = 1, \dots, n$ be denoted by $d_i(k)$. Then from Eq. (8) we have

$$g_i(k) \approx \frac{u_i(k)}{d_i(k)}. \qquad (10)$$

Define

$$\eta_i(k) \triangleq \frac{u_i(k)}{d_i(k)\widehat{g}_i(k)}, \qquad i = 1, 2, \dots, n. \qquad (11)$$

From Eq. (10), we can see $\eta_i(k) \approx \frac{g_i(k)}{\widehat{g}_i(k)}$.

If $|\eta_i(k)-1| \geq \delta$ for some i in $\{1, 2, \dots, n\}$, where δ is a tolerance (a small positive number), we assume the workload changed for processor i at the kth sampling period. In this situation, we re-estimate $g_i(k)$ $(i = 1, \dots, n)$ and use the estimate as a predict to $g_i(k+1)$, denoted by $\widehat{g}_i(k+1)$:

$$\begin{aligned} \widehat{g}_i(k+1) &= \frac{u_i(k)}{d_i(k)}, \\ \widehat{G}(k+1) &\triangleq \text{diag}(\widehat{g}_1(k+1), \dots, \widehat{g}_n(k+1)). \end{aligned} \qquad (12)$$

Note that the estimation of $g_i(k+1)$ in Eq. (12) is computed from Eq. (8) and Eq. (10) by assuming the noise vector $v(k)$ has small deviation.

5.2 G-estimation Based on RLS Approach

At the kth sampling period, when the measured utilization $u(k)$ is close to the setpoint u^s enough such that $|\eta_i(k)-1| < \delta$ for $i = 1, 2, \dots, n$, we assume the workload does not change. Suppose that the workload stays unchanged in an interval starting from the lth sampling period to the kth sampling period ($l < k$), i.e.,

$G(l)=\ldots=G(k)$, instead of using the heuristic approach to define $\widehat{G}(k+1)$ (see Eq. (12)), we prefer solving the following LS problem and set $\widehat{G}(k+1)$ as the solution:

$$\min_{G} \sum_{j=l}^{k} \|u(j) - G*d(j)\|_2^2, \tag{13}$$

where $G \triangleq \mathrm{diag}(g_1, \ldots, g_n) \in \mathbb{R}^{n\times n}$, and $d(j)$ (as defined in Eq. (9)) for $j = l, \ldots, k$ have been determined. To solve the LS problem (13), we define $g \triangleq [g_1, \ldots, g_n]^T \in \mathbb{R}^n$ and $D(j) \triangleq \mathrm{diag}(d_1(j), \ldots, d_n(j)) \in \mathbb{R}^{n\times n}$. Then we can rewrite the LS problem (13) as

$$\min_{g} \left\| \begin{bmatrix} u(l) \\ \vdots \\ u(k-1) \\ u(k) \end{bmatrix} - \begin{bmatrix} D(l) \\ \vdots \\ D(k-1) \\ D(k) \end{bmatrix} g \right\|_2^2 . \tag{14}$$

It is easy to show that the solution to this LS problem can be written as

$$\widehat{g}(k+1) = S(k)^{-1} * w(k),$$

where

$$S(k) \triangleq \sum_{j=l}^{k} D(j)^2 = S(k-1) + D(k)^2,$$
$$w(k) \triangleq \sum_{j=l}^{k} D(j)*u(j) = w(k-1) + D(k)*u(k).$$

Note that the diagonal matrix $S(k-1)$ and the vector $w(k-1)$ were obtained at the $(k-1)$th sampling period. So the computation for finding $\widehat{g}(k+1)$ can be done in a recursive way, known as the RLS approach, and is very efficient. The pseudocode of G-estimator is illustrated in Algorithm 2 in the supplementary document. In the implementation of the above algorithm, actually we do not use two-dimension arrays to store those diagonal matrices for storage efficiency.

6 Utilization Regulator

The utilization regulator finds the optimal manipulated variables F and r by solving the lease squares problem in Eq. (6). We use an alternating direction method to solve this problem. At the kth sampling period, we first fix F in the LS problem (6) to be $F^{(1)}(k)$, where, if $k = 1$, $F^{(1)}(k) = I$ ($I \in \mathbb{R}^{n\times n}$ is the unit matrix), else $F^{(1)}(k)=F(k-1)$ ($F(k-1)$ is F obtained at the $(k-1)$th sampling period). By fixing F to be $F^{(1)}(k)$, we solve the least squares problem (6), where only r is unknown, leading to the solution $r^{(1)}(k)$. Then we fix r in (6) to be $r^{(1)}(k)$ and solve (6), where only F is unknown, leading to the solution $F^{(2)}(k)$. We alternatively fix one variable to find the other, and this iteration process continues until the difference between u^s and $\widehat{G}(k+1)F^{(j)}(k)Er^{(j)}(k)$ is less then a specific threshold ϵ or when the maximum number of iterations is reached. Details about how to find $r^{(j)}(k)$ and $F^{(j+1)}(k)$ for $j = 1, 2, \ldots.$ are given in Section 2.3 of the supplementary document.

7 Performance Evaluation

We implemented a Java-based event-driven simulator in J-Sim [24] to simulate the distributed real-time systems. The utilization control scheme for the distributed real-time systems is implemented as a simulator in MATLAB (R2009a). The communication between MATLAB simulator and J-Sim is via system calls. In each sampling period, the utilization of the processors, the miss deadline ratio of the end-to-end tasks, the power consumption of the system as well as the computing time of the utilization control scheme are computed and used as the criteria to evaluate the utilization control performance. Details of the experimental setup is shown in Section 3.1 and 3.2 of the supplementary document.

Our goal for the performance evaluation lies in two folds. On one hand, we would like to test the necessity of using the frequency scaling to enforce the desired CPU utilization. Therefore, we use EUCON [1] as a baseline since EUCON is a utilization control scheme which adopts task rate adaptation exclusively. Besides EUCON, we use the asynchronous task rate and processor frequency scaling scheme for CPU utilization control proposed in [4] as another baseline, and we denote it by AsyRF. We apply the workload SIMPLE (as shown in Fig. 1 and Table 1) and conduct the simulations to compare the CPU utilization control performance of SyRaFa, AsyRF and EUCON. The workload SIMPLE is set so that the CPU utilization setpoint can never be reached by tuning the task rates exclusively. In this way, we can easily test whether the frequency scaling in SyRaFa or AsyRF is necessary for CPU utilization control. On the other hand, we need to verify the robustness of SyRaFa against workload variations and CPU measurement noise. Moreover, whether SyRaFa outperforms AsyRF in CPU utilization control should be tested. Towards this end, we use AsyRF as the baseline and use the workload MEDIUM to evaluate the utilization control performance of SyRaFa and AsyRF. To make fair comparison, we adopt the MEDIUM workload configuration proposed in [4] and [1], where four processors execute 12 tasks with totally 25 subtasks. Among the 12 tasks, there are 8 end-to-end tasks and 4 local tasks (tasks T_9 to T_{12}). The execution time of every subtask T_{ij} follows a uniform distribution in $[1, 50]$. The task rate range $\mathcal{R}_i$ for each T_i contains 10 discrete values. We set $\mathcal{R}_i$ so that when each processor runs at the highest processor frequency, the maximum CPU utilization is 0.5 (corresponding to the highest task rates of all the tasks) and the minimum CPU utilization is 0.1 (corresponding to the lowest task rates of all the tasks). The CPU utilization setpoint vector for the four processors is $u^s = [0.7348, 0.7348, 0.7286, 0.7348]^T$.

7.1 Experiment I:SIMPLE and Steady Workload

In Experiment I, we use SIMPLE workload as shown in Fig. 1 and Table 1 to conduct simulations. The workload-variation matrix is set as a constant matrix $G = \mathrm{diag}(1, 1)$

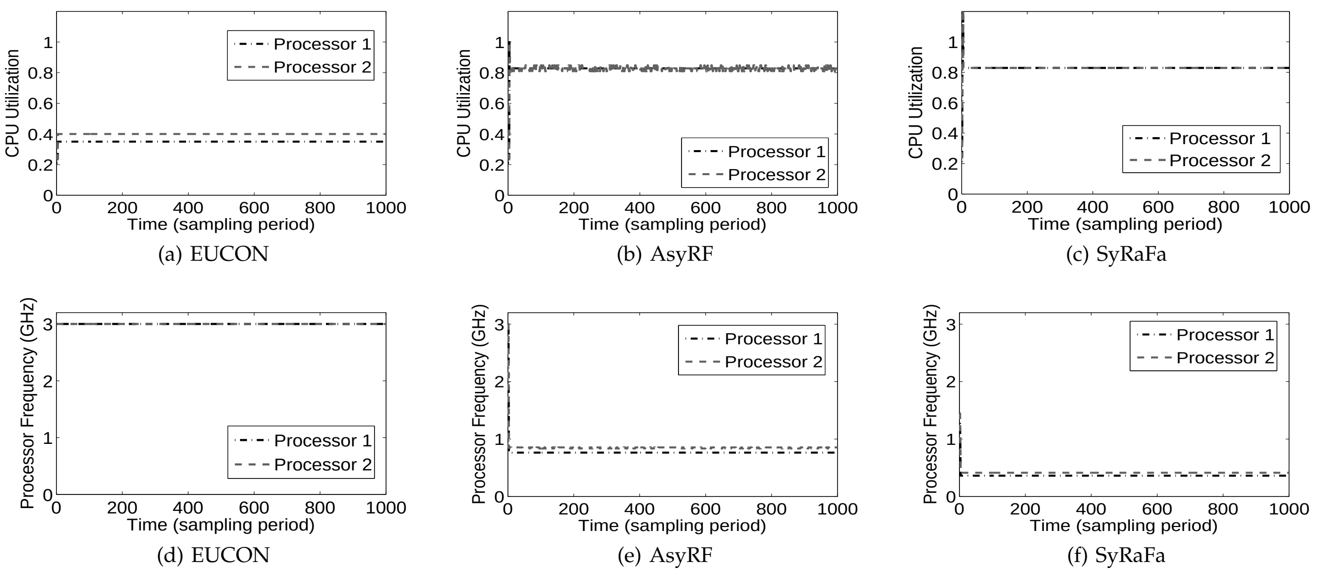


Fig. 4: The Performance of EUCON, AsyRF and SyRaFa in Experiment I

TABLE 1: SIMPLE Workload Configuration

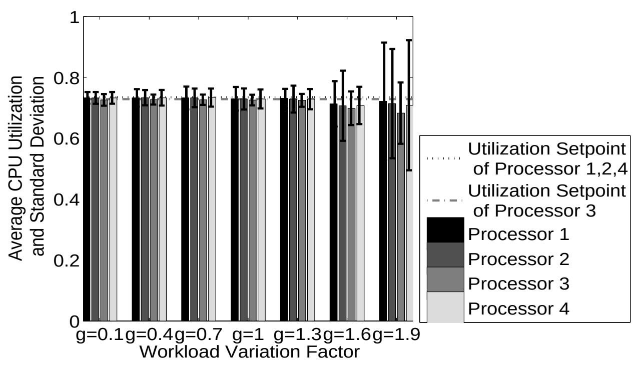
P_i	T_{ij}	c_{ij}	$\mathcal{R}_i$	$r_i(0)$	$u_q^s, q = 1, 2$
P_1	T_{11}	35	$10^{-4} \times \{14, 18, 22, 26, 30, 34, 38, 42, 46, 50\}$	0.0030	0.8284
P_1	T_{21}	35	$10^{-4} \times \{14, 18, 22, 26, 30, 34, 38, 42, 46, 50\}$	0.0030	
P_2	T_{22}	35			
P_2	T_{31}	45	$10^{-4} \times \{11, 15, 20, 24, 28, 33, 37, 41, 46, 50\}$	0.0033	

in each run, which facilitates us to compare the performance of EUCON, AsyRF and SyRaFa in the steady state without workload estimation error. In EUCON, the processor frequencies are set as the highest CPU frequency 3GHz while the task rates $r_i, i = 1 \ldots, m$ are adjusted dynamically online. The utilization control performance of EUCON is shown in Fig. 4(a). The utilizations of the two processors are controlled at around 0.38 and 0.4 respectively, which fail to meet the utilization setpoint 0.8284. In contrast to EUCON, AsyRF and SyRaFa add the processor frequency scaling to the CPU utilization control, and their performance are shown in Figs. 4(b) and (c). Both AsyRF and SyRaFa can enforce the utilization setpoint by dynamically adjusting the processor frequencies, which compensates the control error due to exclusive task rate adaptation. As we can see from Figs. 4(d),(e) and (f), the processor frequencies are lower in AsyRF and SyRaFa than the processor frequencies in EUCON, showing that the CPU frequencies of the processors are scaled down to enforce the utilization setpoint in SyRafa and AsyRF. From the results in Experiment I, we can see frequency scaling is necessary for CPU utilization control.

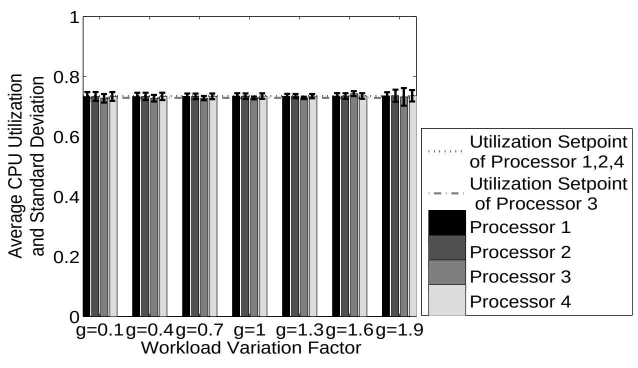
7.2 Experiment II: MEDIUM and Steady Workload

In Experiment II, we use MEDIUM, a more complex workload configuration than SIMPLE, to test the utilization control performance of SyRaFa and AsyRF under different workload configurations. We conduct seven groups of simulations and use steady workload in each group of simulations, which can be achieved by setting the load variation factor g in the load variation matrix $G = g * \mathrm{diag}(1, 1, 1, 1)$ as constant for each group of simulations. Each group of simulations include two runs with each run containing 1000 sampling periods. The utilization control schemes in the two runs are SyRaFa and AsyRF, respectively. To test the performance of SyRaFa and AsyRF under different steady workload configurations, we set g as 0.1, 0.4, 0.7, 1, 1.3, 1.6, 1.9 in the seven groups of simulations separately.

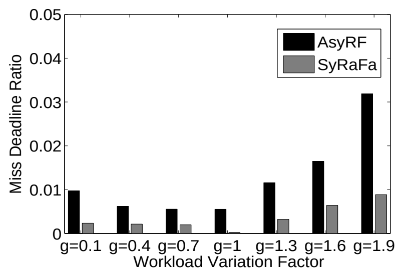
For each group of simulations, we compute the average value and the standard deviation of each processor's utilization during the runtime (1000 sampling periods). Moreover, the average values of the miss deadline ratio, the power consumption and the control time in each run are also calculated. The performance results are shown in Fig. 5. From Figs. 5(a) and (b), we can see both AsyRF and SyRaFa can keep the average CPU utilization of each processor near the setpoint when $g \leq 1.3$. When $g = 1.6$ and $g = 1.9$, the average utilization of each processor under AsyRF cannot track its setpoint properly. AsyRF results in a utilization control error around 0.05 while the proposed scheme SyRaFa can enforce the utilization setpoints. In the seven groups of simulations, the standard deviation of the processor utilization increases with the growth of the workload estimation error (either with the increase of g when workload is underestimated $g > 1$ or with the decrease of g when the workload is overestimated $g < 1$). However, SyRaFa results in much smaller standard deviation of the processor utilization than AsyRF. Compared with AsyRF, SyRaFa provides stable utilization control performance throughout the seven runs with precise setpoint-tracking result. The



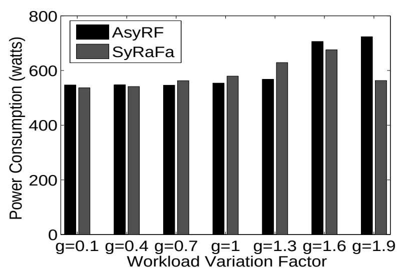
(a) AsyRF



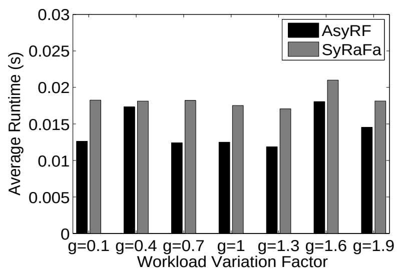
(b) SyRaFa



(c) Average Utilization Comparison



(d) Average Miss Deadline Ratio Comparison



(e) Average Runtime Comparison

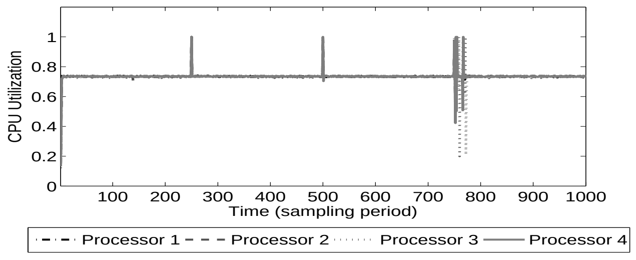
Fig. 5: The Performance of SyRaFa and AsyRF in Experiment II

miss deadline ratios under SyRaFa and AsyRF are shown in Fig. 5(c), where SyRaFa results in a smaller average miss deadline ratio than AsyRF. Both schemes give a miss deadline ratio lower than 0.03, showing that more than 97% of the real-time tasks in the distributed systems can meet their end-to-end deadlines. For the distributed systems with soft real-time tasks, a miss deadline ratio less than 0.03 can provide good QoS. As mentioned above, the underestimation and overestimation to the workload will degrade the control performance in certain extent. As a consequence, the miss deadline ratio when $g > 1$ or $g < 1$ is a little bit larger than that when $g = 1$ under both schemes (SyRaFa and AsyRF). The average power consumptions in the simulations are shown in Fig. 5(d), where SyRaFa and AsyRF cause similar power consumptions. The average control times taken by SyRaFa and AsyRF are shown in Fig. 5(e). SyRaFa has a slightly longer average control time than AsyRF. However, the average control time in SyRaFa is under 0.025s in MATLAB time, which is a small value in comparison with the sampling period (usually several minutes) and does not cause large computation overhead for online implementation.

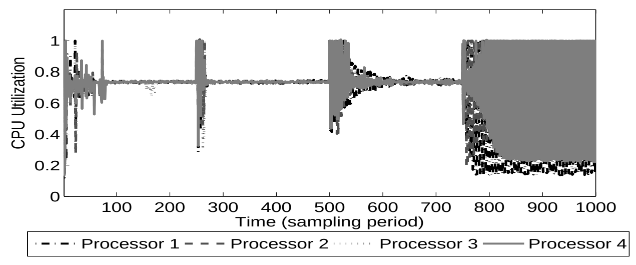
7.3 Experiment III:MEDIUM and Dynamic Workload

In Experiment III, we use MEDIUM workload and test the performance of SyRaFa and AsyRF under dynamic workload and measurement noise. Each run in the simulation contains 1000 sampling periods, we set $g = 0.5$ during 0-249 sampling periods, $g = 1$ during 250-499 sampling periods, $g = 1.5$ during 500-749 sampling periods and $g = 2$ during 750-1000 sampling periods. Therefore, there are sudden changes of the workload at 250th, 500th, 750th sampling periods respectively. Moreover, we add uniformly distributed random noise in the range of $[0, 0.01]$ to the CPU utilization measured from the J-Sim simulator. As we can see from Figs. 6(a) and (b), CPU utilization overshoot occurs when the workload changes. This is because both SyRaFa and AsyRF are feedback-based schemes, the adjustment to the current manipulated variables are based on the estimation to the workload in the previous sampling period, which results in one sampling period control delay. However, after the sudden changes of the workload, SyRaFa takes less sampling periods than AsyRF to make each processor's utilization converge to its setpoint. AsyRF performs badly in handling the workload variations if the workload is underestimated ($g > 1$). When $g = 1.5$, AsyRF takes more than 100 sampling periods to make the CPU utilization u_i fall into the range of $(1 \pm 0.05)u_i^s$. When $g = 2$, the CPU utilization of each processor vibrates around the setpoint and does not converge. In contrast, SyRaFa enforces the CPU utilization setpoints throughout the runtime. The miss deadline ratio and the power consumption in each run under the two schemes are shown in Figs. 6(c-f). SyRaFa gives a much smaller average miss deadline ratio (0.0061) than AsyRF (0.0240). The average power consumptions of the system under SyRaFa (598.6570Watts) and AsyRF (556.2954Watts) are similar.

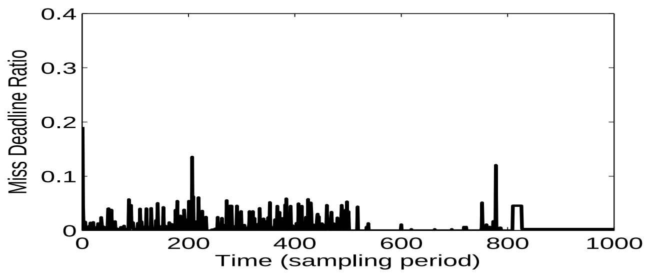
We summarize the average values of the processor utilization, miss deadline ratio, power consumption and



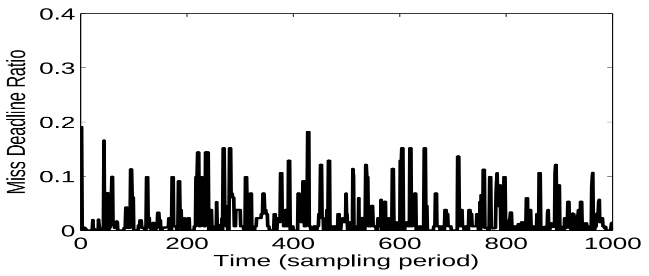
(a) CPU Utilization by SyRaFa



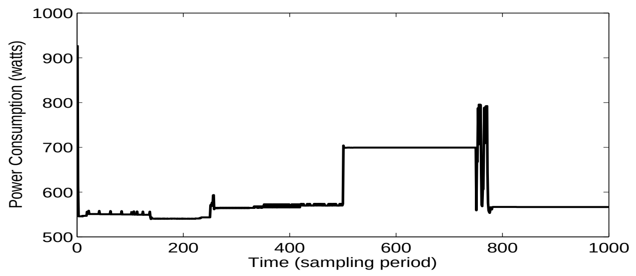
(b) CPU Utilization by AsyRF



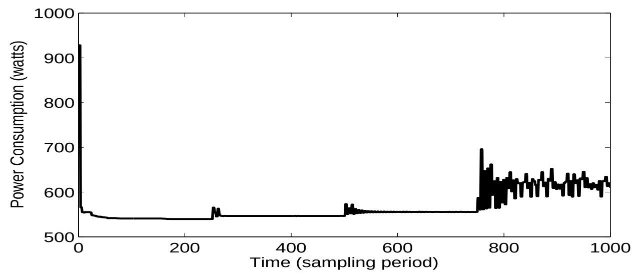
(c) Miss Deadline Ratio by SyRaFa



(d) Miss Deadline Ratio by AsyRF



(e) Power Consumption by SyRaFa



(f) Power Consumption by AsyRF

Fig. 6: The Performance of SyRaFa and AsyRF in Experiment III

the control time in each sampling period during the runtime in the above experiments, which is shown in Table 2 in the supplementary documents. Besides the above experiments, we test the performance of SyRaFa and AsyRF for processor utilization control using a workload setting in the large-scale distributed real-time system. The experiment results are shown in Section 3.3 in the supplementary documents.

8 Conclusions

Both task rates and processor frequencies contribute to the CPU utilizations of the processors in the distributed real-time systems. In order to keep the processor utilizations close to the setpoint in a distributed real-time system, we need to tune task rates and processor frequencies together for efficient processor utilization control. Existing method uses feedback control to tune the two manipulated variables separately in two control loops with asynchronous sampling periods. We have provided a novel scheme called SyRaFa that simultaneously adjusts the two variables to enforce desired CPU utilizations of the processors. In SyRaFa, a new strategy, which takes the utilization measurement noise into account, is used to estimate the workload by predicting the load-variation matrix online. Then we compute the optimal manipulated variables (task rates and processor frequencies) by solving a bilinear mixed-discrete least squares problem. Simulation results have demonstrated that the SyRaFa scheme is effective especially in the situation with large workload uncertainties and frequent workload fluctuations.

Acknowledgement

The authors would like to thank the referees for providing valuable suggestions. This work was supported in part by NSERC of Canada Grants 217191-07 and 341823-07, FQRNT of Quebec Grant 2010-NC-131844, CFI Leaders Opportunity Fund 23090, and National Science Foundation of USA Award #1117664.

References

[1] 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, pp. 550–561, 2005.

[2] X. Wang, Y. Chen, C. Lu, and X. Koutsoukos, "On Controllability and Feasibility of Utilization Control in Distributed Real-Time Systems," in *Proceedings of 19th Euromicro Conference on Real-Time Systems (ECRTS)*, pp. 103–12, 2007.

[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, pp. 996–1009, 2007.

[4] X. Wang, X. Fu, X. Liu, and Z. Gu, "Power-Aware CPU Utilization Control for Distributed Real-Time Systems," in *Proceedings of 15th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pp. 233–242, 2009.

[5] J. Yao, X. Liu, Z. Gu, X. Wang, and J. Li, "Online Adaptive Utilization Control for Real-Time Embedded Multiprocessor Systems," *Journal of Systems Architecture (Elsevier)*, vol. 56, no. 9, pp. 463–473, 2010.

[6] Y. Fu, H. Wang, C. Lu, and R. S. Chandra, "Distributed Utilization Control for Real-Time Clusters with Load Balancing," in *Proceedings of the 27th IEEE International Real-Time Systems Symposium (RTSS)*, pp. 137–146, 2006.

[7] J. Sun and J. Liu, "Synchronization Protocols in Distributed Real-Time Systems," in *Proceedings of the 16th International Conference on Distributed Computing Systems*, pp. 38–45, 1996.

[8] J. Goossens, S. Funk, and S. Baruah, "Priority-Driven Scheduling of Periodic Task Systems on Multiprocessors," *Real-Time Systems*, vol. 25, no. 2, pp. 187–205, 2003.

[9] J. Anderson, V. Bud, and U. Devi, "An EDF-Based Restricted-Migration Scheduling Algorithm for Multiprocessor Soft Real-Time Systems," *Real-Time Systems*, vol. 38, no. 2, pp. 85–131, 2008.

[10] K. Årzén, A. Robertsson, D. Henriksson, M. Johansson, H. Hjalmarsson, and K. Johansson, "Conclusions of the ARTIST2 Roadmap on Control of Computing Systems," *ACM SIGBED Review*, vol. 3, no. 3, pp. 11–20, 2006.

[11] T. Abdelzaher, J. Stankovic, C. Lu, R. Zhang, and Y. Lu, "Feedback Performance Control in Software Services," *IEEE Control Systems Magazine*, vol. 23, no. 3, pp. 74–90, 2003.

[12] A. Goel, Walpole, and M. Shor, "Real-Rate Scheduling," in *in proceedings of the 10th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pp. 434–441, 2004.

[13] J. Stankovic, C. Lu, S. Son, and G. Tao, "The Case for Feedback Control Real-Time Scheduling," in *Proceedings of the 11th Euromicro Conference on Real-Time Systems (ECRTS)*, pp. 11–20, 1999.

[14] X. Wang, C. Lu, and C. Gill, "FCS/nORB: A Feedback Control Real-Time Scheduling Service for Embedded ORB Middleware," *Microprocessors and Microsystems (Elsevier)*, vol. 32, no. 8, pp. 413–424, 2008.

[15] X. Koutsoukos, R. Tekumalla, B. Natarajan, and C. Lu, "Hybrid Supervisory Utilization Control of Real-Time Systems," in *Proceedings of 11th IEEE Real Time and Embedded Technology and Applications Symposium (RTAS)*, pp. 12–21, 2005.

[16] Y. Chen, C. Lu, and X. Koutsoukos, "Optimal Discrete Rate Adaptation for Distributed Real-Time Systems," in *Proceedings of the 28th IEEE International Real-Time Systems Symposium (RTSS)*, pp. 181–192, 2007.

[17] H. Aydin and D. Zhu, "Reliability-Aware Energy Management for Periodic Real-Time Tasks," *IEEE Transactions on Computers*, vol. 58, no. 10, pp. 1382–1397, 2009.

[18] J. Chen, C. Hung, and T. Kuo, "On the Minimization for the Instantaneous Temperature for Periodic Real-Time Tasks," in *Proceedings of the 13th IEEE Real Time and Embedded Technology and Applications Symposium, (RTAS)*, pp. 236–248,2007.

[19] R. Xu, R. Melhem, and D. Mossé, "Energy-Aware Scheduling for Streaming Applications on Chip Multiprocessors," in *Proceedings of the 28th IEEE Real-Time System Symposium (RTSS)*, pp. 25–38, 2007.

[20] S. Saewong and R. Rajkumar, "Practical Voltage-Scaling for Fixed-Priority RT-systems," in *Proceedings of the 9th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pp. 106–114, 2003.

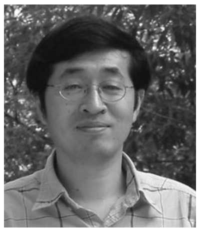
[21] H. Aydi, P. Mejía-Alvarez, D. Mossé, and R. Melhem, "Dynamic and Aggressive Scheduling Techniques for Power-Aware Real-Time Systems," in *Proceedings of the 21th IEEE Real-Time Systems Symposium*, pp. 95–105, 2001.

[22] Y. Zhu and F. Mueller, "Feedback EDF Scheduling of Real-Time Tasks Exploiting Dynamic Voltage Scaling," *Real-Time Systems*, vol. 31, no. 1, pp. 33–63, 2005.

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

[24] A. Sobeih, M. Viswanathan, D. Marinov, and J. Hou, "J-Sim: An integrated Environment for Simulation and Model Checking of Network Protocols," in *IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2007, pp. 1–6.

Xi Chen received the B.E. degree in 2007 from Northwestern Polytechnical University, Xi'an, China, and is currently a PhD candidate of Computer Science at McGill University, Montreal, Quebec, Canada. Her research interests are in the areas of real-time scheduling, feedback resource allocation in distributed systems.



Xiao-Wen Chang received his B.S. and M.S. degrees in Computational Mathematics from Nanjing University, Nanjing, China, in 1986 and 1989, respectively, and his Ph.D. degree (Dean's Honor List) in Computer Science from McGill University, Montreal, Quebec, Canada, in 1997. He is currently an Associate Professor in the School of Computer Science at McGill University. His research interests are in the area of scientific computing, with particular emphasis on numerical linear algebra and its applications. He has published over forty papers in refereed journals.

Xue Liu is an associate professor in the School of Computer Science at McGill University. He received his Ph.D. in Computer Science from the University of Illinois at Urbana-Champaign in 2006. He received his B.S. degree in Mathematics and M.S. degree in Automatic Control both from Tsinghua University, China. He has also worked as the Samuel R. Thompson Associate Professor in the University of Nebraska-Lincoln and HP Labs in Palo Alto, California. His research interests are in computer networks and communications, smart grid, real-time and embedded systems, cyber-physical systems, data centers, and software reliability. Dr. Liu has been granted 1 US patent and filed 4 other US patents, and published more than 150 research papers in major peer-reviewed international journals and conference proceedings, including the Year 2008 Best Paper Award from IEEE Transactions on Industrial Informatics, and the First Place Best Paper Award of the ACM Conference on Wireless Network Security (WiSec 2011).