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

Xi Chen, Xiao-Wen Chang, and Xue Liu

1 Notation Table

We summarize the notations used in the paper in Table 1 for clarity.

TABLE 1: Notations in SyRaFa Scheme

T_i, $i = 1, \ldots, m$	the ith end-to-end task in the system
T_{ij}, $j = 1, \ldots, t_i$	the jth subtask of T_i
c_{ij}	the estimated execution time of T_{ij} when the processor holding T_{ij} runs at the highest processor frequency
r_i	the task rate of T_i and its subtasks T_{ij}, $j = 1, \ldots, t_i$
$\mathcal{R}_i = \{\gamma_{i1}, \ldots, \gamma_{il}\}$	the predefined set of r_i
P_q, $q = 1, \ldots, n$	the qth processor in the system
S_q	the set of subtasks T_{ij} held by P_q
f_q	the normalized CPU frequency of P_q
$[f_{min}, 1]$	the range for f_q
u_q	CPU utilization of P_q
u_q^s	CPU utilization setpoint of P_q
g_q	workload variation factor
$G = \text{diag}(g_1, \ldots, g_n) \in \mathbb{R}^{n \times n}$	load-variation matrix
$F = \text{diag}(\frac{1}{f_1}, \ldots, \frac{1}{f_n}) \in \mathbb{R}^{n \times n}$	frequency-inverse matrix
$E = (e_{ij}) \in \mathbb{R}^{m \times n}$	subtask allocation matrix
$r = [r_1, \ldots, r_m]^T \in \mathbb{R}^m$	task rate vector
$u = [u_1, \ldots, u_n]^T \in \mathbb{R}^n$	processor utilization vector
$v \in \mathbb{R}^n$	utilization measurement noise
T_s	the length of sampling period
$k = 1, \ldots, \frac{\text{(Total Runtime)}}{T_s}$	index of the sampling period
$E(k-1), F(k-1), G(k)$	E, F and G at the kth sampling period
$\widehat{G}(k+1) \in \mathbb{R}^{m \times n}$	predict to $G(k+1)$ computed at the end of kth sampling period

2 SyRaFa Framework

2.1 SyRaFa Working Principles

The utilization control procedures are summarized as follows:

- The utilization monitor of each processor measures its current utilization $u_i(k)$ and sends it to G-estimator.
- G-estimator uses the measurement of all the processors' utilization $u(k)$, the historical utilization measurements $\{u(1), u(2), ..., u(k-1)\}$ and the setpoint vector u^s to predict $\widehat{G}(k+1)$ based on the G-estimator algorithm illustrated in Section 5 (Online Estimation of Utilization Model) of the main document, which is then sent to the utilization regulator.
- Based on u^s and $\widehat{G}(k+1)$, the utilization regulator solves the LS problem in Eq. (6) in the main document and provides $r(k)$ and $F(k)$ simultaneously. Details on the utilization control scheme are given in Section 6 (Utilization Regulator) of the main document and in Section 2.3 of this supplementary document. The manipulated variables $r(k)$ and $F(k)$ are then sent to $rate\,modulator$ and $frequency\,modulator$ on each processor.
- The $rate\,modulator$ and $frequency\,modulator$ adjust the task rates and processor frequencies according to $r(k)$ and $F(k)$.

This utilization control procedures are invoked in each sampling period. The pseudocode of SyRaFa scheme is given in Algorithm 1.

Algorithm 1 SyRaFa scheme

1: **for** $k = 0$ to (Total Runtime)/T_s **do** {$k = 0$ is the initialization stage}
2: **if** $k > 0$ **then**
3: Apply $F(k-1)$, $r(k-1)$ to the distributed real-time system
4: Get measured utilization $u(k)$ by utilization monitor
5: **end if**
6: Call G-estimator in Algorithm 2 to find $\widehat{G}(k+1)$
7: Call the utilization control scheme in Algorithm 3 to find $F(k)$, $r(k)$
8: **end for**

2.2 G-estimator Algorithm

The pseudocode of G-estimator is illustrated in Algorithm 2.

Algorithm 2 G-estimator (given the threshold δ)

if $k=0$ **then** {Initialization stage}
  Initialize $\widehat{G}(1)$ as a unit matrix
  Set $l=1$ {l is the starting period of the RLS approach}
else {During runtime}
  Compute $d(k)=F(k-1)*E*r(k-1)$
  Compute $\eta_i(k)=\frac{u_i(k)}{d_i(k)\widehat{g}_i(k)}$ for $i=1,2,\ldots,n$
  if $|\eta_i(k)-1|\geq\delta$ for some i in $\{1,2,\ldots,n\}$, **then**
    Compute $\widehat{g}_i(k+1)=\frac{u_i(k)}{d_i(k)}$ for $i=1,2,\ldots,n$
    Form $\widehat{G}(k+1)=\mathrm{diag}(\widehat{g}_1(k+1),\ldots,\widehat{g}_n(k+1))$
    Set $l=k+1$
  else
    Form $D(k)=\mathrm{diag}(d_1(k),\ldots,d_n(k))$
    if $k=l$ **then**
      Compute $S(k)$, $w(k)$ by:

$$S(k)=D(k)^2$$
$$w(k)=D(k)*u(k)$$

    else
      Compute $S(k)$, $w(k)$ by:

$$S(k)=S(k-1)+D(k)^2$$
$$w(k)=w(k-1)+D(k)*u(k)$$

    end if
    $\widehat{g}(k+1)=S(k)^{-1}*w(k)$
    Form $\widehat{G}(k+1)=\mathrm{diag}(\widehat{g}_1(k+1),\ldots,\widehat{g}_n(k+1))$
  end if
end if

2.3 Utilization Regulator

The utilization regulator finds the optimal manipulated variables F and r by solving the lease squares problem in Eq. (6) in the main document. As introduced in the main document, we use an alternating direction method to solve this problem. At the kth sampling period, we first fix F in the LS problem (6) in the main document 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) in the main document, where only r is unknown, leading to the solution $r^{(1)}(k)$. Then we fix r in problem (6) in the main document to be $r^{(1)}(k)$ and solve the least squares problem, 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.

2.3.1 Fix F and Find r

Suppose that $F^{(j)}(k)$ has been found in the jth iteration in the kth sampling period. We now fix F in the LS problem (6) in the main document to be $F^{(j)}(k)$ and solve the LS problem

$$\min_r \|u^s-\widehat{G}(k+1)*F^{(j)}(k)*E*r\|_2 \quad (1)$$
$$\text{subject to} \quad r_i\in\mathcal{R}_i,\ 1\leqslant i\leqslant m,$$

resulting in the solution $r^{(j)}(k)$.

For each i, the set $\mathcal{R}_i$ includes several discrete values. Thus (1) can be referred to as a discrete LS problem. This problem is very similar to a box-constrained integer LS (ILS) problem, where each component of the unknown vector is an integer lying in an interval (see, e.g., [1] and [2]). Algorithms for box-constrained ILS problems can be extended to solve (1). Let $B\triangleq\widehat{G}(k+1)*F^{(j)}(k)*E\in\mathbb{R}^{n\times m}$. The rank of B depends on the number of processors, the number of tasks, and the allocation of the subtasks on different processors. If $m\leq n$, we assume B has full column rank (i.e., $\mathrm{rank}(B)=m$), otherwise, we assume B has full row rank (i.e., $\mathrm{rank}(B)=n$). The above assumption is usually satisfied in practice. Even if we have a rank deficient case, our approach can still handle it. Here we assume that B is of full row rank, as it is a more complicated and more pervasive situation. One such application is a distributed real-time system which has more tasks than processors. The solution to the full-row-rank B case can easily be modified to solve the problem with full-column-rank B.

To solve the discrete LS problem (1), we have slightly modified the search algorithm given in [1], which solves a full-row-rank integer LS problem. In the following we introduce the main ideas of the algorithm.

We first compute the QR factorization of B: $B=Q*R$, where $Q\in\mathbb{R}^{n\times n}$ is orthogonal and $R\in\mathbb{R}^{n\times m}$ is upper trapezoidal. Without loss of generality, we assume the diagonal entries of R are positive. With $\bar{u}^s\triangleq Q^T*u^s$, the discrete LS problem in Eq. (1) is transformed to

$$\min_r \|\bar{u}^s-R*r\|_2, \text{ subject to } r_i\in\mathcal{R}_i,\ 1\leqslant i\leqslant m. \quad (2)$$

We should find a constant β such that the solution $r^{(j)}(k)$ is included in the hyper-ellispoid:

$$\|\bar{u}^s-R*r\|_2^2<\beta^2. \quad (3)$$

Then, the tree search process given in [1] can be slightly modified to determine r. To reduce the computational cost in each iteration step, we do not solve the discrete LS problem exactly. Instead, we set the initial β to be ∞ and use the first point we find in the search process as $r^{(j)}(k)$. For simplicity we do not reorder the columns of the matrix as suggested in [1]. For more details, refer to [1].

2.3.2 Fix r and Find F

After we obtain $r^{(j)}(k)$, we fix r in the LS problem (6) in the main document to $r^{(j)}(k)$ and solve the following

LS problem

$$\min_{F} \|u^s - \widehat{G}(k+1)*F*E*r^{(j)}(k)\|_2 \quad (4)$$
$$\text{subject to } f_{\min} \leqslant f_q \leqslant 1,\ 1 \leqslant q \leqslant n,$$

resulting in the solution $F^{(j+1)}(k)$, whose qth diagonal entry is denoted by $1/f_q^{(j+1)}(k)$.

Now we show how to solve (4). Let $b \triangleq \widehat{G}(k+1)*E*r^{(j)}(k) \in \mathbb{R}^n$. Since both $\widehat{G}(k+1)$ and F are diagonal matrices,

$$\begin{aligned}\widehat{G}(k+1)*F*E*r^{(j)}(k) &= F*\widehat{G}(k+1)*E*r^{(j)}(k)\\ &= Fb = [b_1/f_1, b_2/f_2, \ldots, b_n/f_n]^T.\end{aligned}$$

Thus, for the objective function of the LS problem in Eq. (4) we have

$$\|u^s - \widehat{G}(k+1)*F*E*r^{(j)}(k)\|_2 = \Big(\sum_{q=1}^{n}(u_q^s - b_q/f_q)^2\Big)^{1/2}.$$

Note that both u_q^s and b_q are positive. It is easy to verify that the optimal f_q satisfies

$$f_q^{(j+1)}(k) = \begin{cases} b_q/u_q^s & \text{if } b_q/u_q^s \in [f_{\min}, 1]\\ f_{\min}, & \text{if } b_q/u_q^s < f_{\min}\\ 1, & \text{if } b_q/u_q^s > 1 \end{cases}, \ 1 \leqslant q \leqslant n.$$

The scheme of utilization regulator is shown in Algorithm 3.

Algorithm 3 Utilization Control Scheme (given the threshold ϵ and the maximum number of iterations z)

1: **if** $k = 0$ **then** {Initialization stage}
2: Set $F^{(1)}(k) = I$
3: **else** {During runtime}
4: Set $F^{(1)}(k)$ as $F(k-1)$
5: **end if**
6: Set $j = 1$
7: **while** do
8: Fix F as $F^{(j)}(k)$ to solve the discrete LS problem in Eq. (1), resulting $r^{(j)}(k)$
9: **if** $\|u^s - \widehat{G}(k+1)*F^{(j)}(k)*E*r^{(j)}(k)\|_2 < \epsilon$ or $j > z$ **then**
10: Break
11: **else**
12: Fix r as $r^{(j)}(k)$ to solve the LS problem in Eq. (4), resulting $F^{(j+1)}(k)$
13: $j = j + 1$
14: **end if**
15: **end while**
16: Set $F(k) = F^{(j)}(k)$ and $r(k) = r^{(j)}(k)$

3 Performance Evaluation

3.1 Experimental Setup

We implemented a Java-based event-driven simulator in J-Sim [3] to simulate the distributed real-time systems, where we simulate the sequential release of the subtasks in each end-to-end task according to Phase Modification synchronization protocol [4], the rate monotonic scheduling of the subtasks on each processor and the dynamic CPU frequency scaling of each processor. The utilization control scheme for the distributed real-time systems is implemented as a simulator in MATLAB (R2009a). Both J-Sim and MATLAB run on the same physical machine (Intel dual core quad CPU at 2.66GHz and 3GB RAM) with Linux operating system. We use system calls to let the MATLAB simulator communicate with the J-Sim simulator. The sampling period for the utilization control is chosen so that the task with the longest end-to-end deadline can be invoked several times during one sampling period. In the kth sampling period for $k = 1, 2, \ldots$, the control scheme implemented by MATLAB computes the optimal manipulated variables and enables J-Sim to run, where the control scheme implemented by MATLAB sends the manipulated variables as parameter to the distributed system implemented in J-Sim. J-Sim then simulates the operation of the distributed real-time system in the kth sampling period with the task rates and processor frequencies configured according to the manipulated variables received from MATLAB. At the end of the kth sampling period for $k = 1, 2, \ldots$, J-Sim simulator saves the measured variables during this sampling period in a text file and pauses. The measured variables saved by J-Sim simulator in the kth sampling period include the miss deadline ratio of the tasks $\rho(k)$, the measured processor utilization vector $u(k)$ and the power consumption of the system $W(k)$. Then the utilization controller implemented by MATLAB reads these results $(u(k), \rho(k), W(k))$ from the text file and computes the optimal manipulated variables $r(k)$ and $F(k)$. $r(k)$ and $F(k)$ will be used by J-Sim in the next sampling period, namely the $(k+1)$th sampling period. The above procedures repeat in every sampling period until the end of the runtime.

The relative end-to-end deadline of each task T_i is $\frac{t_i}{r_i(k)}$, where t_i is the number of subtasks in T_i. The relative end-to-end deadline of T_i is divided into the relative deadlines of its subtasks with each subtask having a relative deadline $\frac{1}{r_i(k)}$. Since rate monotonic scheduling is applied to schedule all the subtasks located on each processor, the schedulable utilization bound of rate monotonic scheduling [5] is used as the setpoint on each processor, which can be computed as

$$u_i^s = m_i(2^{\frac{1}{m_i}} - 1), \quad (5)$$

where m_i is the number of subtasks on P_i. By enforcing the utilization setpoint of each processor, we would like to provide a small miss deadline ratio even under severe workload uncertainties. In the kth sampling period for $k = 1, 2, \ldots$, the J-Sim simulator records the response time of every end-to-end task instance. Moreover, the J-Sim simulator computes the miss deadline ratio $\rho(k)$, which is the ratio of the number of the instances that fail to meet their end-to-end deadlines to the total number

of instances released in the sampling period.

The workload variations in the distributed systems can be simulated by tuning the load-variation matrix G. Each diagonal entry in G, denoted by g_i, reflects the deviation of the actual utilization of the processor to its estimated value. The g_i larger than 1 indicates an underestimation to the CPU utilization of P_i while the g_i less than 1 indicates an overestimation. For example, in order to set g_i as 2, we can adjust the actual execution time of each subtask located on Processor P_i twice as large as its estimated value.

To verify the effectiveness of the utilization control in reducing the power consumption of the distributed systems, the J-Sim simulator also simulates the power consumption of the processors. Since the power consumption of each processor usually has a cubic relation with the normalized processor frequency [6], [7], the simulator computes the power consumption of the whole distributed system as follows

$$W(k) = \sum_{q=1}^{n} (W_I + \alpha f_q^3), \tag{6}$$

where $W(k)$ is the total electricity consumption of the distributed systems at the kth sampling period, W_I is the static power consumption of each processor at the idle mode and α is the coefficient (in Watts). In our simulation, we assume the CPU frequency of each processor can be adjusted in the range from 0.3GHz to 3GHz, hence the normalized processor frequency f_q is within $[f_{min}, 1]$ with $f_{min} = 0.1$. According to the power profile proposed in [7], we set W_I in Eq. (6) as 134Watts, and α as 98.01Watts .

To evaluate the computation overhead of SyRaFa, we would like to get the time taken by the centralized utilization controller to compute the manipulated variables. Since the controller is implemented in MATLAB, we use the MATLAB functions *tic* and *toc* to compute the MATLAB time taken by the controller to obtain the manipulated variables in each sampling period, denoted by $t_c(k)$.

In our experiments, δ in Algorithm 2 is taken to be 0.1. z in Algorithm 3 is taken to be 3 and ϵ in Algorithm 3 is taken to be 10^{-5}.

3.2 Performance Metrics

We use the following four performance metrics to evaluate the effectiveness of the schemes including SyRaFa, EUCON and AsyRF in utilization control.

- Average utilization tracking error e,

$$e \triangleq \frac{1}{M} \sum_{k=1}^{M} \sum_{i=1}^{n} (u_i(k) - u_i^s)^2, \quad M \triangleq \frac{T}{T_s}. \tag{7}$$

- Miss deadline ratio ρ,

$$\rho \triangleq \frac{1}{M} \sum_{k=1}^{M} \rho(k), \quad M \triangleq \frac{T}{T_s}. \tag{8}$$

- Average power consumption W,

$$W \triangleq \frac{1}{M} \sum_{k=1}^{M} W(k), \quad M \triangleq \frac{T}{T_s}. \tag{9}$$

- Average control time t_c,

$$t_c \triangleq \frac{1}{M} \sum_{k=1}^{M} t_c(k), \quad M \triangleq \frac{T}{T_s}. \tag{10}$$

Note that T and T_s in Eqs. (7-10) are the total runtime and the sampling period respectively. The smaller e, ρ, W and t_c, the more effective of the utilization control scheme.

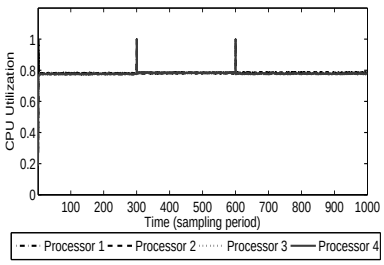
3.3 Experiment IV:LARGE and Dynamic Workload

We would like to see the performance of the proposed scheme SyRaFa in comparison with AsyRF in large-scale distributed real-time systems, hence we use workload LARGE to evaluate the performance of the two schemes. In workload LARGE, there are 12 processors executing 24 tasks with totally 36 subtasks. The execution time of each subtask follows a uniform distribution in $[1, 50]$. The task rate range $\mathcal{R}_i$ for each T_i contains 10 discrete values, and the task rate ranges ($\mathcal{R}_i$ for $i = 1, \ldots, m$) are chosen so that when the processor frequencies stay at their highest CPU frequencies, the maximum processor utilization is 0.5 and the minimum processor utilization is 0.1. The utilization setpoint u_q^s is 0.7798 for $q = 1, \ldots, 12$.

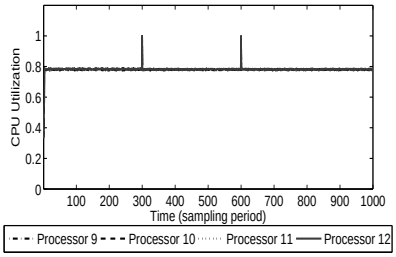
In Experiment IV, we use LARGE workload and test the performance of SyRaFa and AsyRF under the dynamic workload and measurement noise. Each run contains 1000 sampling periods, we set $g = 0.5$ in 1-299 sampling periods, $g = 1$ in 300-599 sampling periods and $g = 1.5$ in 600-1000 sampling periods. Moreover, 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. 1(a-f), while SyRaFa can control the utilization of the 12 processors in the system to track the utilization setpoints, AsyRF fails to enforce the utilization setpoints of the 12 processors during 300-1000 sampling periods due to the workload variations and the existence of measurement noise. Especially, processor 4,5,9 experience utilization saturation (utilization stays at 1) during 600-1000 sampling periods. As a result, while SyRaFa provides a small miss deadline ratio (around 0.04) during the runtime, AsyRF produces a high miss deadline ratio up to 0.2 during 300-1000 sampling periods, as shown in Figs. 1(g,h). SyRaFa causes smaller power consumption of the distributed real-time systems than AsyRF, as shown in Figs. 1(i,j).

3.4 Performance Comparison of SyRaFa and AsyRF

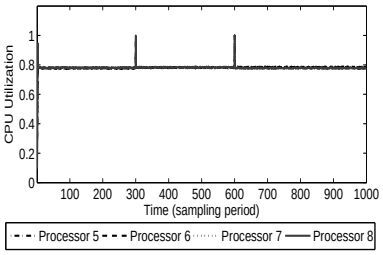
We summarize the values of the performance metrics obtained in Experiment I, III and IV in Table 2. The values of the performance metrics obtained in Experiment II are shown in Fig. 5 in the main document.



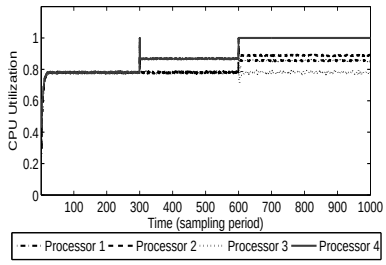
(a) CPU Utilization by SyRaFa (Processors 1-4)



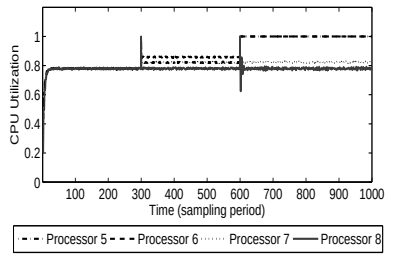
(b) CPU Utilization by SyRaFa (Processors 5-8)



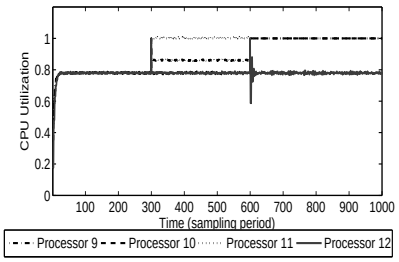
(c) CPU Utilization by SyRaFa (Processors 9-12)



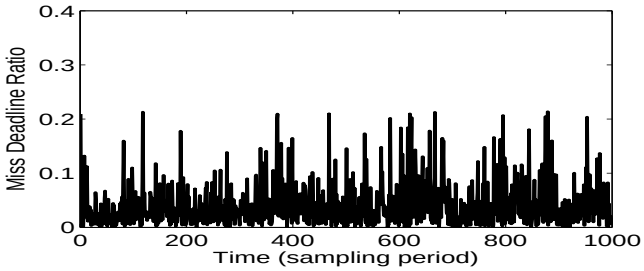
(d) CPU Utilization by AsyRF (Processors 1-4)



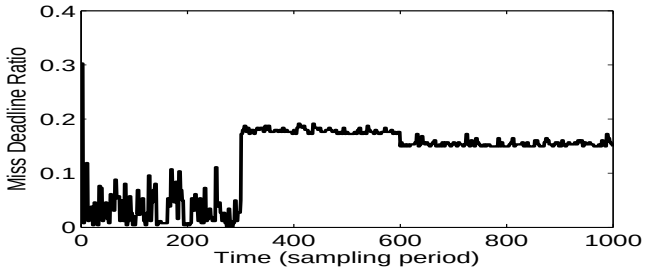
(e) CPU Utilization by AsyRF (Processors 5-8)



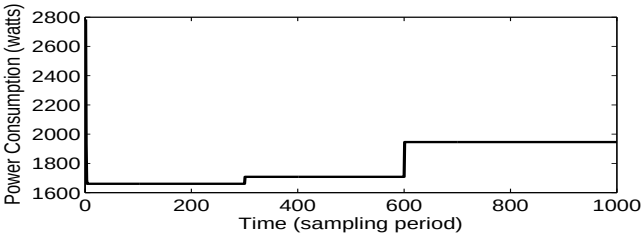
(f) CPU Utilization by AsyRF (Processors 9-12)



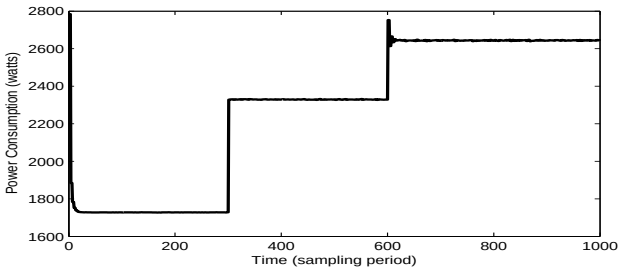
(g) Miss Deadline Ratio by SyRaFa



(h) Miss Deadline Ratio by AsyRF



(i) Power Consumption by SyRaFa



(j) Power Consumption by AsyRF

Fig. 1: The Performance of SyRaFa and AsyRF in Experiment IV

From the results, we observe that SyRaFa can provide a better CPU utilization setpoint-tracking performance with smaller e (see Eq. (7)). Moreover, SyRaFa results in a smaller miss deadline ratio than AsyRF during the four experiments. In the first three experiments, SyRaFa and AsyRF result in similar power consumption of the distributed system. In experiment IV, SyRaFa results in 1739.1Watts power consumption while AsyRF results in 2279.0Watts power consumption. Although the average control time in SyRaFa is slightly larger than the average control time in AsyRF, SyRaFa scheme is efficient for online applications. In general, SyRaFa provides better CPU utilization control performance than AsyRF. SyRaFa also provides better QoS to the real-time tasks in the distributed systems than AsyRF. This demonstrates that synchronous adjustment to the manipulated variables for CPU utilization control outperforms the asynchronous adjustment to the manipulated variables in the decoupled control loops.

TABLE 2: Performance Comparison of SyRaFa, AsyRF and EUCON in Three Experiments

Exp.No.	Scheme	e	W	ρ	t_c
I	EUCON	0.4134	464.0200	0.0008	0.0059
	SyRaFa	0.0012	268.6656	0.0009	0.0024
	AsyRF	0.0026	272.4084	0.0035	0.0062
III	SyRaFa	0.0080	598.6570	0.0061	0.0249
	AsyRF	0.4487	556.2954	0.0240	0.0113
IV	SyRaFa	0.0063	1739.1	0.0370	0.0415
	AsyRF	0.6880	2279.0	0.1261	0.0210

4 Discussion

4.1 Efficient Utilization Control Scheme

To reduce the computational overhead of the control scheme, instead of reconfiguring the task rates and CPU frequencies in each sampling period, we can keep the same system configurations until the workload changes. To be specific, at the end of each sampling period k, $k = 1, 2, \ldots$, if $|\eta_i(k)-1| \leq \delta$ ($\eta_i(k)$ is defined in Eq. (11) in the main document and is also given in Algorithm 2) for $i = 1, 2, \ldots, n$, where δ is a tolerance (a small positive number), we suppose that the workload does not change in the kth sampling period, and we predict that the workload will not change in the $(k+1)$th sampling period. In this situation, instead of computing the manipulated variables for the next sampling period, namely r and F in the $(k+1)$th sampling period, we will keep r and F unchanged. We only compute the manipulated variables at the end of the kth sampling period if $|\eta_i(k)-1| > \delta$ for some i in $\{1, 2, \ldots, n\}$. Using this strategy, it will reduce the computational overhead of the control scheme. It will also reduce the implementation cost of the distributed system since it avoids adjusting the CPU frequencies or task rates in every sampling period.

4.2 Scalability Issue

We have done extensive experiments to test the performance of SyRaFa and AsyRF in different workload configurations. We find that in the large-scale distributed systems (with processors more than 12 and total tasks more than 24), both SyRaFa and AsyRF schemes experience the scalability difficulty and cannot always guarantee the good utilization control performance. The scalability problem experienced by SyRaFa is due to the following reason. The alternating direction method for computing $F(k)$ and $r(k)$ at the end of each sampling period stops either when the optimal manipulated variables ($F(k)$ and $r(k)$) are found or when the maximum number of iterations is reached. In large-scale systems, the optimal manipulated variables may not be found within the maximum number of iterations, resulting in poor utilization control performance.

To solve the scalability problem, we could use a decentralized utilization control scheme that decomposes the large-scale system into several small-scale subsystems and then arranges a local utilization controller for each subsystem. The scalability problem will be investigated in our future work.

References

[1] X.-W. Chang and X. Yang, "An Efficient Tree Search Decoder with Column Reordering for Underdetermined MIMO Systems," in *Proceedings of IEEE Global Telecommunications Conference,(GLOBECOM)*, pp. 4375–4379, 2007.

[2] X.-W. Chang and Q. Han, "Solving Box-Constrained Integer Least Squares Problems," *IEEE Transactions on Wireless Communications*, vol. 7, no. 1, pp. 277–287, 2008.

[3] 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.

[4] 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.

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

[6] Y. Chen, A. Das, W. Qin, A. Sivasubramaniam, Q. Wang, and N. Gautam, "Managing Server Energy and Operational Costs in Hosting Centers," *ACM SIGMETRICS Performance Evaluation Review*, vol. 33, no. 1, pp. 303–314, 2005.

[7] A. Gandhi, M. Harchol-Balter, R. Das, and C. Lefurgy, "Optimal Power Allocation in Server Farms," in *Proceedings of the 11th international joint conference on Measurement and modeling of computer systems*, pp. 157–168,2009.