Proceedings of the
44th IEEE Conference on Decision and Control, and
the European Control Conference 2005
Seville, Spain, December 12-15, 2005

Multi-Resource Allocation Control for Fair QoS Management in Real-Time Systems

Fumiko Harada†, Toshimitsu Ushio†, Yukikazu Nakamoto‡
†: Graduate School of Engineering Science, Osaka University,
{harada@hopf., ushio@}sys.es.osaka-u.ac.jp
‡: Graduate School of Applied Informatics, University of Hyogo, nakamoto@ai.u-hyogo.ac.jp

Abstract— **In this paper, we propose an adaptive resource control method for multiple resource real-time systems. Execution results of applications are evaluated as a QoS level and a fair QoS management is an important issue. The fair QoS level depends on the number of active applications and their characteristics so that it may change dynamically due to the current states of the real-time systems. The proposed controller activates at discrete times and updates resource allocations to achieve the optimal fair QoS level. Moreover, the algorithms used in the controller solve the update in $O(mn)$ time per its activation. We derive sufficient conditions for achievement of the fair resource allocation. Simulation experiments show that a fair resource allocation can be achieved under the conditions.**

I. Introduction

In real-time systems, overload conditions bring up the significant degrade of system response predictability and performance[12]. To avoid overload conditions in soft real-time systems such as multimedia computing systems, QoS (Quality of Service) control methods have been proposed, where resource requirements are reduced according to a QoS level allocated to each task. Flexible applications are often used to improve QoS as long as its computation time and resources increase[1]. In flexible applications, the QoS level indicates a satisfaction level of users for an execution result of released jobs. When each flexible application in a system increases the QoS level simultaneously, however, the system becomes overload conditions. To avoid the conditions, arbitrating the QoS levels of the competitive applications are required. There are several researches to arbitrate the competitive applications or tasks in real-time systems. Abdelzaher *et al.* have proposed a method where the acceptable QoS levels of tasks in overload conditions are described *a priori* and the system degrades the QoS levels of the tasks based on the QoS level description at overload conditions[4]. However, this study does not consider dynamic negotiation of QoS levels of tasks. Rajkumar *et al.* have proposed the QoS-based resource allocation model to solve problems when applications in real-time systems have simultaneous access to multiple resources[2].

A resource allocation based on the QoS level may cause unfairness. When allocated resource are decreased with the same ratio for all tasks to avoid overload conditions or to arbitrate the QoS levels of competitive tasks, the quality of services of tasks varies and the deviation of the QoS levels could occur. Several QoS management methods for prevention of unfairness of QoS levels have been proposed. In [3], a system utility function for fair-sharing is described as maximizing the lowest QoS level among the task set.

In such QoS management methods, a resource allocation problem is described by a multi-dimensional nonlinear programming problem for static environments. Thus these methods cannot be applied into dynamically changing environments where the task set, workload, and/or available resource amount are dynamically varying. To adapt quality of services in dynamically varying real-time systems, the control theory is utilized to manage resource allocations to tasks[6]. In [7] and [8], feedback control methods are adopted to maintain CPU ratios and deadline ratios with specified values. Feedback control QoS adaptation in Internet Servers are proposed in [5]. In [9], both a feedback mechanism for the bandwidth allocation and an admission control mechanism are utilized for the QoS management in radio networks. In [10], a feedback mechanism is applied to control robot applications according to the temporal requirement changes. Diao *et al.* proposed an LQR control method for load balancing of computing systems[13]. The cost function is modeled by specifying the weight matrices based on the impact of control actions and transient load imbalances.

As a feedback control architecture for fairness of QoS levels among the current task set, we have proposed a QoS adaptation control method where the updated CPU utilization factor converges a fair CPU allocation and fair QoS level is achieved[11]. In this paper, we will extend it and propose a novel QoS adaptation control system to achieve the fairness of QoS levels for the multi-resource real-time environment.

Conventional feedback scheduling architectures require reference values *a priori*, which may requires recalculation of them when active tasks change. In the recalculation, solving nonlinear equations requires $O(n^2)$ computation time, where n is the number of the active tasks, and characteristics of QoS levels of all tasks must be known. However, the proposed controller searches the optimal fair QoS level online without increase of computation time in the controller. The controller takes $O(mn)$ time complexity, where m and n are the number of resources and tasks, respectively, which means efficient computation time. Moreover, though several feasible fair resource allocations may exist under multi-resource environments, the QoS adaptation controller can provide the best resource allocation in sight of the efficient resource usage.

This paper is organized as follows: In Section II, we will describe considered real-time systems and define fairness of QoS level in a task set. A QoS adaptation control for fair resource allocation is introduced in Section III. The architecture of this control, formulation of control objective, and a control rule are shown. We analyse this system and show a sufficient condition to achieve fair QoS in Section IV. The result of simulation experiment of our method is shown in Section V. Finally, we will conclude the paper in Section VI.

II. Real-time system model

A. Resource and task set

In this paper, we consider a real-time system with n independent tasks $\{\tau_1, \tau_2, \dots, \tau_n\}$ and m resources $\{\mathcal{R}_1, \mathcal{R}_2, \dots, \mathcal{R}_m\}$. Each resource $\mathcal{R}_j$ has a finite capacity R_j and can be shared with the tasks, either temporally or spatially. A specified amount of every resource is needed for execution of each task with the acceptable lowest QoS level and it is improved by additional resource allocations.

Each task τ_i releases a job periodically or aperiodically. The job is executed and completed with using an allocated portion of each resource. When each job released by τ_i accomplishes a QoS level $QoS_i \in \mathbb{R}$, we say that the QoS level of τ_i is QoS_i.

Let $r_{ij}(\in \mathbb{R}^+)$ be the portion of resource $\mathcal{R}_j$ allocated to task τ_i. QoS_i is improved by allocating more amount of resources. The relationship between the resource allocation $\{r_{i1}, \dots, r_{im}\}$ and the QoS level QoS_i is described by a non-decreasing *resource consumption function*. Its detail will be discussed in Section II-C.

Each task τ_i has the minimum and maximum QoS requirement, $QoS_i^{\min}$ and $QoS_i^{\max}$, respectively, where $QoS_i^{\min}$ represents the worst QoS level acceptable to τ_i and $QoS_i^{\max}$ the optimal QoS level τ_i requires. Thus, we have $QoS_i^{\min} \le QoS_i \le QoS_i^{\max}$ for each task τ_i.

In order to achieve $QoS_i^{\min}$ or $QoS_i^{\max}$, τ_i needs the minimum or maximum resource requirement $r_{ij}^{\min}$ and $r_{ij}^{\max}$ for each resource $\mathcal{R}_j$, respectively.

B. Fairness of QoS levels

In real-time systems, resource allocations may cause unfairness. For example, task τ_1 is executed with its maximum QoS level $QoS_1^{\max}$ while task τ_2 is executed with its minimum QoS level $QoS_2^{\min}$ for the same resource allocation $r_{1j} = r_{2j}$ for each resource $\mathcal{R}_j$. In general, the range $[QoS_i^{\min},\ QoS_i^{\max}]$ of the QoS level QoS_i is individually associated with task τ_i, and different each other. To evaluate fairness, we introduce a normalized QoS level Q_i for task τ_i as follows:

$$Q_i := \frac{QoS_i - QoS_i^{\min}}{QoS_i^{\max} - QoS_i^{\min}}. \tag{1}$$

Note that Q_i is monotonically increasing according to $r_{i1}, \dots, r_{im}$, whose range is $[0, 1]$. It represents a satisfaction rate of execution results in the sense which $Q_i = 0$ means that a job released by τ_i is executed with its minimum QoS level $QoS_i^{\min}$ while $Q_i = 1$ means that it is executed with its maximum QoS level $QoS_i^{\max}$.

The relative importance of each task τ_i is denoted by $w_i \in \mathbb{R}^+$. The larger w_i is, the more critical or important among $\{\tau_1, \tau_2, \dots, \tau_n\}$ τ_i is. Then we define weighted fairness of the QoS level as follows[11]:

Definition 1: Suppose that each task $\tau_i (i = 1, 2, \dots, n)$ releases a job sharing each resource $\mathcal{R}_j (j = 1, 2, \dots, m)$ with r_{ij}. Let Q_i and w_i be the normalized QoS level and the importance. Then, we will say that a fair resource allocation is achieved by r_{ij} if the following equation holds:

$$\frac{1}{w_1} Q_1 = \frac{1}{w_2} Q_2 = \cdots = \frac{1}{w_n} Q_n. \tag{2}$$

Moreover, $Q^f = Q_i / w_i$ is called the "fair QoS level".
In the following, Q_i will be simply called the QoS level of τ_i.

Remark 1: Since we assume that each Q_i is real-valued and continuously varied as r_{ij} changes, the fair QoS level which satisfies (2) always exists.

C. Resource utilization and QoS level

As described in the previous section, the relationship between resource allocation $\{r_{i1}, r_{i2}, \dots, r_{im}\}$ and the corresponding achievable QoS level Q_i is described by the resource consumption function $\phi_{ij} : [0, 1] \to [0, R_j]$. $\phi_{ij}(q)$ represents the amount of resource $\mathcal{R}_j$ needed to achieve QoS level q. Thus, To achieve QoS level q of τ_i, τ_i requires its resource allocation $\{\phi_{i1}(q), \dots, \phi_{im}(q)\}$.

When task τ_i can use r_{ij} for resource $\mathcal{R}_j$, its QoS level is determined as follows: For simplicity, we consider $m = 2$. As shown in Fig. 1, let $Q_i^{\text{high}} = \phi_{i1}^{-1}(r_{i1})$ and $Q_i^{\text{low}} = \phi_{i2}^{-1}(r_{i2})$. In order for a job released by τ_i to be completed with Q_i^{high}, a required amount for $\mathcal{R}_2$ is $\phi_{i2}(Q_i^{\text{high}})$. However, since $r_{i2} < \phi_{i2}(Q_i^{\text{high}})$, r_{i2} is insufficient to achieve Q_i^{high}. An achievable QoS level is Q_i^{low} since $r_{ij} \ge \phi_{ij}(Q_i^{\text{low}}) (j = 1, 2)$. Thus, the QoS level of the execution is evaluated as Q_i^{low} and the utilized amount of resource $\mathcal{R}_1$ is $\phi_{i1}(Q_i^{\text{low}}) < r_{i1}$. On the other hand, all allocated amount of $\mathcal{R}_2$ is utilized for the execution. In the following, the utilized amount of resource $\mathcal{R}_j$ for the execution of a job released by τ_i will be denoted by r_{ij}^{act}. From the above discussion, the following equations hold in general:

$$Q_i = \min_j \phi_{ij}^{-1}(r_{ij}),\ \ r_{ij}^{\text{act}} = \phi_{ij}(Q_i). \tag{3}$$

r_{ij}^{act} represents the actual resource utilization of resource $\mathcal{R}_j$ while r_{ij} represents the allocated (or reserved) portion. Note that each task τ_i releases a job completed with the highest QoS level under allocated resources. Thus, there exists a *fully-consumed resource* $\mathcal{R}_j$ for task τ_i such that $r_{ij} = r_{ij}^{\text{act}}$.

Remark 2: Under a single-resource environment ($m = 1$), allocated resource $\mathcal{R}_1$ is fully consumed for execution of tasks. Thus, the following equation always holds:

$$Q_i = \phi_{i1}(r_{i1}),\ \ r_{i1}^{\text{act}} = r_{i1}. \tag{4}$$

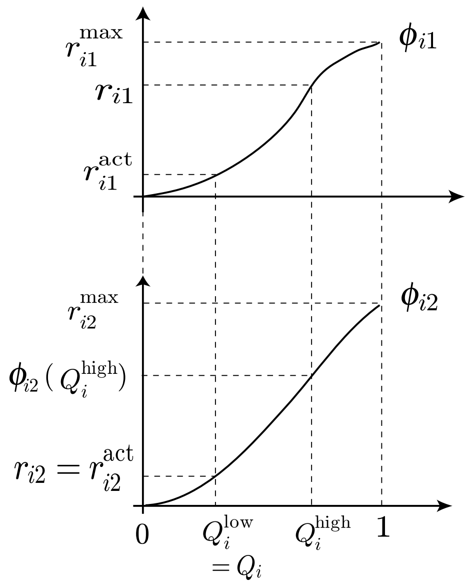

Fig. 1. The relationship between resource utilizations and QoS level.

D. Control objective

In this paper, we consider the following optimization problem:

Problem 1: Maximize the fair QoS level subject to the following resource constraints: for each j,

$$\sum_{i=1}^{n} r_{ij} \le R_j. \tag{5}$$

To solve this problem, we will propose a resource allocation controller based on the errors between tasks' QoS levels and their average. Since Q_i and r_{ij} dynamically vary according to the resource allocation control, we denote their current value during the time interval $[t_k, t_{k+1})$ as $Q_i(k)$ and $r_{ij}(k)$, where t_k represents the k-th controller activation time.

Moreover, for simplicity, we make some assumptions as follows:

- $w_i = 1$ for each task τ_i without loss of generality.
- Each ϕ_{ij} is a monotonically increasing and diffeomorphic function.
- $r_{ij}^{\min} = 0$ for all $i = 1, 2, \dots, n$ and $j = 1, 2, \dots, m$. Note that, when $r_{ij}^{\min} > 0$, we can replace $R_j - \sum_{i=1}^{n} r_{ij}^{\min}$ and $r_{ij}^{\max} - r_{ij}^{\min}$ with R_j and $r_{ij}^{\max}$, respectively.
- There exist constants h_{ij} and H_{ij} for all ϕ_{ij} such that

$$0 < h_{ij} \le \frac{d\phi_{ij}}{dQ_i} \le H_{ij} < \infty, \tag{6}$$

which implies $0 < \frac{1}{H_{ij}} \le \frac{d\phi_{ij}^{-1}}{dr_i} \le \frac{1}{h_{ij}} < \infty$.

III. QoS adaptation control

In this section, we introduce a QoS adaptation controller to achieve a fair resource allocation, with extension of our previous work[11]. In Subsection III-A, we describe the architecture of the control system. In Subsection III-B, a control rule are introduced.

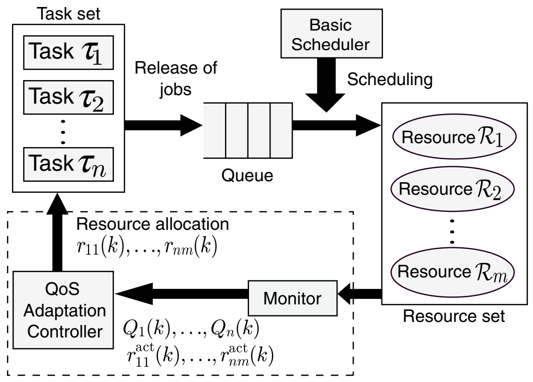

Fig. 2. Real-time systems with QoS adaptation control.

A. Architecture

Shown in Fig. 2 is our proposed QoS adaptation control architecture for fair sharing, which consists of a basic scheduler, a QoS adaptation controller, and a monitor.

The monitor evaluates a normalized QoS level of each completed job and feeds it back to the QoS adaptation controller.

The QoS adaptation controller activates at discrete times $t_1, t_2, \dots,$ and allocates the resources to each task with searching a fair QoS level on-line. This allocation is based on the normalized QoS levels fed back from the monitor. A control rule used in the QoS adaptation controller will be dealt with in the next subsection.

The basic scheduler schedules each released job based on the resource portions allocated to the corresponding task by the QoS adaptation controller. It works with a specified algorithm for resource allocation (e.g., for CPU resource, Earliest Deadline First, Rate Monotonic, and so on).

The controller activates at discrete times $t_k (k = 1, 2, \dots)$. When the QoS adaptation controller activates at time $t = t_k$, it updates $r_{11}(k), \dots, r_{nm}(k)$ based on QoS levels $Q_1(k-1), \dots, Q_n(k-1)$, which are fed back from the monitor. Every job of τ_i which is released in time $[t_k, t_{k+1})$ is allocated $r_{ij}(k)$ of Resource $\mathcal{R}_j$. Its QoS level is uniquely determined according to (3).

B. Control rule

Conventional feedback control rules such as PID control are based on an error between a reference value and its current value in general. Many studies on applications of the control theory to real-time systems assume that the reference values (depending on control specifications) are given *a priori* [6], [7]. On the other hand, we have proposed a control system where an unknown fair QoS level is searched on-line according to the following control rule[11]:

$$r_{i1}(k+1) = r_{i1}(k) + \alpha(\overline{Q}(k) - Q_i(k)), \tag{7}$$

where

$$\overline{Q}(k) = \frac{1}{n} \sum_{l=1}^{n} Q_l(k). \tag{8}$$

In this method, a new CPU allocation is determined based on the error between each QoS level and the average of all QoS levels and a fair CPU allocation is achieved when the error converges to zero. In this paper, we extend this control rule to achieve a fair resource allocation for the multi-resource environment.

From (3), each available resource is not always consumed fully. To maximize the achievable QoS level as high as possible, we consider the *minimum resource unutilization factor* $\lambda(k)$ for each k given by

$$\lambda(k) = 1 - \max_j \left(\frac{\sum_{i=1}^n r_{ij}^{\text{act}}(k)}{R_j} \right). \tag{9}$$

$\sum_{i=1}^n r_{ij}^{\text{act}}(k)/R_j$ is the actual total utilization factor of resource $\mathcal{R}_j$ for all tasks. If

$$\lambda(k) = 0 \tag{10}$$

holds, there exists at least one bottleneck resource $\mathcal{R}_j$ which is consumed fully by its maximum capacity R_j. This implies that the QoS level Q_i is optimal under the fair resource allocation when both (2) and (10) hold.

We extend the control rule (7) to multi-resource allocations as follows: for each $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, m$,

$$\begin{aligned} r_{ij}(k+1) =& \Big(1 + \alpha\lambda(k)(1 - Q_i(k))\Big) r_{ij}^{\text{act}}(k) \\ &+ \beta \underline{h}_j (\overline{Q}(k) - Q_i(k)), \end{aligned} \tag{11}$$

where

$$\alpha, \beta \in \mathbb{R}^+, \tag{12}$$

$$\underline{h}_j = \min_i h_{ij}. \tag{13}$$

Note that this control rule requires $O(nm)$ computation time, where nm is the number of equations in (11).

Equation (11) can be explained as follows: If $Q_i(k)$ is smaller than $\overline{Q}(k)$, each resource allocation $r_{ij}(k)$ to τ_i is increased, which implies $r_{ij}^{\text{act}}(k+1) > r_{ij}^{\text{act}}(k)$. This means that the updated resource utilizations $r_{i1}(k+1), \ldots, r_{im}(k+1)$ upgrade the new QoS level $Q_i(k+1)$ than the previous one $Q_i(k)$. On the other hand, $r_{ij}(k)$ is decreased if $Q_i(k) > \overline{Q}(k)$. So the error between $Q_i(k)$ and the average gets smaller and smaller, which implies $|\overline{Q}(k) - Q_i(k)|$ converges to zero. When $\lambda(k)$ is positive, there are unused portions in all resources so that QoS levels of all tasks can be improved by allocating the unused portions to them.

When every $\overline{Q}(k) - Q_i(k)$ and $\lambda(k)$ is equal to zero, the control objective is achieved. Note that the following equation holds simultaneously, which is preferable for efficient resource usage:

$$r_{ij}(k) = r_{ij}^{\text{act}}(k). \tag{14}$$

Remark 3: As described in Remark 2 in Subsection II-C, $r_{i1}(k) = r_{i1}^{\text{act}}(k)$ and $\lambda(k) = 0$ for any k in the case where $m = 1$. Then the control rule (11) can be rewritten as (7).

In (11), the selection of appropriate the control parameters α and β is important. Inadequate α or β violate feasibility and accomplishment of fair allocation. They must be selected to satisfy the following two conditions:

- **[Feasibility condition]** Each $r_{ij}(k)(i = 1, 2, \ldots, n,\ j = 1, 2, \ldots, m)$ is positive for every k. Also, Each resource constraint $\sum_{i=1}^n r_{ij}(k) \le R_j$ is guaranteed.
- **[Stability condition]** Every $r_{ij}(k)$ converges to the optimal fair resource allocation, that is a fixed point of the closed-loop system.

IV. Analysis of the closed-loop system

In this section, we show sufficient conditions for which both feasibility and stability condition are guaranteed. For simplicity, we make an assumption on the initial resource allocation:

$$r_{ij}(0) \ge 0, \ \sum_{i=1}^n r_{ij}(0) \le R_j. \tag{15}$$

A. Feasibility condition

If the control parameters α and β are too large, $r_{ij}(k)$ will cause a large variation even if the error $|\overline{Q}(k) - Q_i(k)|$ and $\lambda(k)$ are small, which leads to infeasible situations such as $r_{ij}(k) < 0$ or $r_{ij}(k) > R_j$. Moreover, in order to satisfy resource constraints, the total utilization of resource $\mathcal{R}_j$ should be equal or less than R_j. Thus, the following conditions must hold for every k:

$$r_{ij}(k) \ge 0, \ \sum_{i=1}^n r_{ij}(k) \le R_j. \tag{16}$$

The following lemma guarantees the feasibility of the control rule.

Lemma 1 (Feasibility condition): Equation (16) holds for every k if

$$0 \le \alpha \le 1, \ 0 < \beta \le \frac{n}{n-1}. \tag{17}$$

proof: See [15].

B. Stability condition

In this subsection, we discuss stabilization by the proposed QoS adaptation control, which guarantees achievement of the fair QoS level. The closed-loop system described by (3) and (11) has two fixed points. To discuss stability around the optimal fair allocation point, we show the following fact at first.

Fact 1: The closed-loop system has the following two fixed points:
(I) $r_{ij}(k) = 0$ for all $i = 1, 2, \ldots, n, j = 1, 2, \ldots, m$ (that is every $Q_i(k)$ equals to zero).
(II) $r_{ij}(k) = r_{ij}^f$ such that (2), (10) and (14) hold. Let the corresponding QoS level be Q^f.

The fixed point (I) is unstable. Since $r_{ij}^{\text{act}}(k) = 0$ for all i, j, the first arguments of the right side of (11) equals to zero, regardless of $\lambda(k)$ $(= 1)$. Also, since $Q_i(k) = 0$, every $\overline{Q}(k) - Q_i(k)$ equals to zero. This implies that $r_{ij}(k+1) = 0$ for all $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, m$. However, even if a $r_{ij}(k)$ takes sufficiently small number, $r_{ij}^{\text{act}}(k) > 0$ and $r_{ij}(k+1)$ increases largely, which implies that the fixed point (I) is unstable.

Now we discuss the local stability around the fixed point (II), $r_{ij} = r^f_{ij}$. The *allocation error vector* δr , the *actual utilization error vector* δr^{act}, and the *QoS error vector* δQ are defined as follows:

$$\delta r(k) = \begin{bmatrix} r_{11} - r^f_{11} \\ \vdots \\ r_{1m} - r^f_{1m} \\ \vdots \\ r_{n1} - r^f_{n1} \\ \vdots \\ r_{nm} - r^f_{nm} \end{bmatrix}, \delta r^{\mathrm{act}}(k) = \begin{bmatrix} r^{\mathrm{act}}_{11} - r^f_{11} \\ \vdots \\ r^{\mathrm{act}}_{1m} - r^f_{1m} \\ \vdots \\ r^{\mathrm{act}}_{n1} - r^f_{n1} \\ \vdots \\ r^{\mathrm{act}}_{nm} - r^f_{nm} \end{bmatrix}, \quad (18)$$

$$\delta Q(k) = \begin{bmatrix} Q_1(k) - Q^f \\ \vdots \\ Q_n(k) - Q^f \end{bmatrix}. \quad (19)$$

We denote the *resource consumption error function* ϕ_{ij} by

$$\phi_{ij}(Q_i(k) - Q^f) - r^{\mathrm{act}}_{ij} = \delta\phi_{ij}(\delta Q_i(k)). \quad (20)$$

Let j_i and j_u be the indexes of resources which satisfy

$$r_{ij_i} = r^{\mathrm{act}}_{ij_i}(k), j_u = \arg\left(\max_{1\le j\le m} \frac{\sum_{i=1}^n r^{\mathrm{act}}_{ij}(k)}{R_j} \right), \quad (21)$$

respectively. Using $\delta r_{ij}(k)$, $\delta r^{\mathrm{act}}_{ij}(k)$, and $\delta Q_i(k)$, (11) can be rewritten as follows:

$$\begin{aligned} \delta r_{ij}(k+1) =& \left[1 + \alpha\lambda(k)(1 - Q^f - \delta Q_i(k))\right]\delta r^{\mathrm{act}}_{ij}(k) \\ &+ \beta\underline{h}_j(\overline{\delta Q}(k) - \delta Q_i(k)) \\ &+ \alpha\lambda(k)(1 - Q^f - \delta Q_i(k))r^f_{ij}. \end{aligned} \quad (22)$$

Then the closed-loop system can be linearized around the fixed point (II) as follows:

$$\begin{aligned} \delta r(k+1) =& \left[P_1\tilde{h}_{1j_1} \quad P_2\tilde{h}_{2j_2} \quad \cdots \quad P_n\tilde{h}_{nj_n}\right]\delta r(k) \\ :=& P \times \delta r(k), \end{aligned} \quad (23)$$

where $O_{i,j}$ is the $i \times j$ zero matrix and

$$P_i = \left[O_{nm,j_i-1} \quad C_i \quad O_{nm,m-j_i}\right], \quad (24)$$

$$A_i = \begin{bmatrix} A_{i1} \\ \vdots \\ A_{im} \end{bmatrix}, \ B^i_k = \begin{bmatrix} B^i_{k1} \\ \vdots \\ B^i_{km} \end{bmatrix}, \ C_i = \begin{bmatrix} B^i_1 \\ \vdots \\ B^i_{i-1} \\ A_i \\ B^i_{i+1} \\ \vdots \\ B^i_n \end{bmatrix}, \quad (25)$$

$$\begin{aligned} A_{ij} =& \frac{d\delta\phi_{ij}(0)}{d\delta Q_i} - \frac{\beta\underline{h}_j(n-1)}{n} \\ &- \frac{\alpha}{R_{j_u}}(1 - Q^f)r^f_{ij}\frac{d\delta\phi_{ij_u}(0)}{d\delta Q_i}, \end{aligned} \quad (26)$$

$$B^p_{ij} = \frac{\beta\underline{h}_j}{n} - \frac{\alpha}{R_{j_u}}(1 - Q^f)r^f_{ij}\frac{d\delta\phi_{pj_u}(0)}{d\delta Q_p}, \quad (27)$$

$$\tilde{h}_{ij_i} = \frac{d\delta\phi^{-1}_{ij_i}(0)}{d\delta r_{ij_i}}. \quad (28)$$

TABLE I

Task parameters for simulation.

	R_i	h_{i1}	h_{i2}	h_{i3}	h_{i4}	h_{i5}	h_{i6}
$\mathcal{R}_1$	1	0.63	0.98	0.35	0.94	0.62	0.76
$\mathcal{R}_2$	3	2.8	2.47	1.96	2.44	2.76	2.94
$\mathcal{R}_3$	2	1.85	0.78	1.47	1.30	1.95	1.19

Remark: every h_{ij} is equal to H_{ij}.

If the origin of (23) is asymptotically stable, every $\delta r_{ij}(k)$ converges to zero and the fixed point (II) is asymptotically stable. In order to prove the asymptotic stability, it is sufficient to show that absolute values of the all eigenvalues of P are less than 1. It is easily shown that the eigenvalues of P equal to zero or those of the following matrix P':

$$P' := \begin{bmatrix} A_{1j_1}\tilde{h}_{1j_1} & B^2_{1j_1}\tilde{h}_{1j_1} & \cdots & B^n_{1j_1}\tilde{h}_{1j_1} \\ B^1_{2j_2}\tilde{h}_{2j_2} & A_{2j_2}\tilde{h}_{2j_2} & \cdots & B^n_{2j_2}\tilde{h}_{2j_2} \\ \vdots & \vdots & \ddots & \vdots \\ B^1_{nj_n}\tilde{h}_{nj_n} & B^2_{nj_n}\tilde{h}_{nj_n} & \cdots & A_{nj_n}\tilde{h}_{nj_n} \end{bmatrix}. \quad (29)$$

A stability condition is given by the following lemma.

Lemma 2 (Stability condition): If

$$0 < \alpha \cdot \max_{i,j} \frac{H_{ij}}{\underline{h}_j} \cdot \max_{i,j} \frac{H_{ij}}{R_j} \cdot \min_j \frac{nR_j}{\sum_{i=1}^n h_{ij}} \le \beta \le 1, \quad (30)$$

then all eigenvalues of P' are stable.

Proof: See [15].

C. Selection of α and β

According to Lemmas 1 and 2, (15), (17), and (30) give sufficient conditions for satisfying the control objective. This can be summarized as the following theorem:

Theorem 1 (Fair resource allocation condition): If

$$0 < \alpha \le 1, \quad (31)$$

$$\alpha \cdot \max \frac{H_{ij}}{\underline{h}_j} \cdot \max \frac{H_{ij}}{R_j} \cdot \min \frac{nR_j}{\sum_{i=1}^n h_{ij}} \le \beta \le 1, \quad (32)$$

then the optimal fair resource allocation is achieved by the control rule (11) under the initial condition satisfying (15). For fast convergence of $Q_i(k)$ to Q^f, α and β may be selected as large as possible, where

$$\alpha = \frac{1}{\max \frac{H_{ij}}{\underline{h}_j} \cdot \max \frac{H_{ij}}{R_j} \cdot \min \frac{nR_j}{\sum_{i=1}^n h_{ij}}}, \ \beta = 1. \quad (33)$$

V. Simulation

We show simulation experiments to evaluate the performance of the proposed QoS adaptation control. We consider an independent task set $\{\tau_1, \tau_2, \dots, \tau_6\}$ and a resource set $\{\mathcal{R}_1, \mathcal{R}_2, \mathcal{R}_3\}$. Each ϕ_{ij} is a linear function. Shown in Table I are task parameters and capacities of resource, where $h_{ij} = H_{ij}$.

The control parameters α and β are set to 0.312 and 1, respectively, which satisfy (33).

The simulation results are shown in Figs. 3, 4, and 5. Fig. 3 shows that all $Q_i(k)$ converges to 0.195, which implies

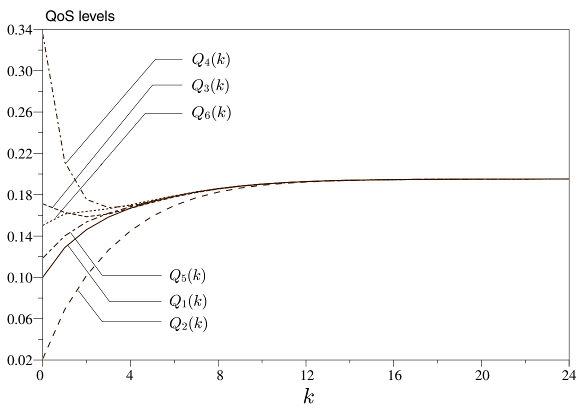

Fig. 3. Behavior of QoS.

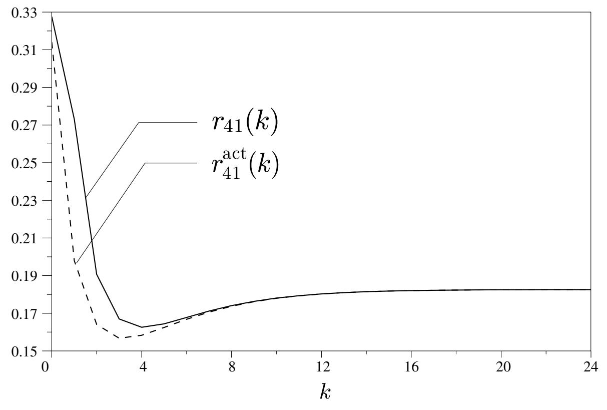

Fig. 5. Behavior of allocated and actual utilization of $\mathcal{R}_4$ of τ_1.

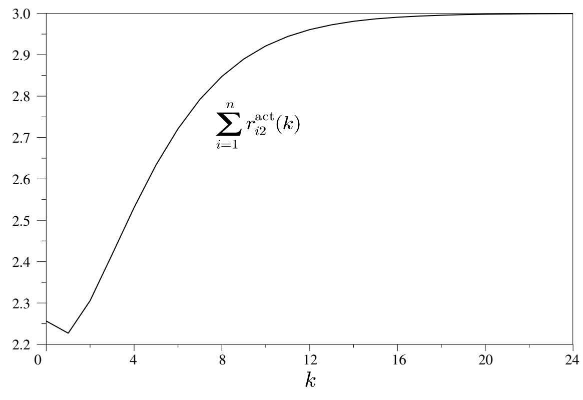

Fig. 4. Behavior of the total utilization of $\mathcal{R}_2$.

that (2) holds. Fig. 4 shows that the total actual utilization $\sum_{i=1}^{n} r_{i2}^{\text{act}}(k)$ equals to R_2 in the steady state, which means that (14) is satisfied. These two Figures show that the fair resource allocation is achieved with the highest QoS level. In sight of (10), every $r_{ij}^{\text{act}}(k)$ goes up to $r_{ij}(k)$ and $r_{ij}^{\text{act}}(k) = r_{ij}(k)$ in the steady state. For example, Fig. 5 shows that $r_{41}^{\text{act}}(k)$ is equal to $r_{41}^{\text{act}}(k)$ in the steady state, while $r_{41}^{\text{act}}(k)$ is less than $r_{41}(k)$ in transient state. Thus the control objective is accomplished, which means the proposed method can perform the fair resource allocation.

VI. Conclusions

In this paper, we proposed a control method to achieve a fair resource allocation based on QoS levels in multi-task and multi-resource systems. Based on the errors between current QoS levels and their average, the proposed QoS adaptation controller searches the desirable QoS level and allocates resources to each task in order to achieve the level. This control is very simple so that it takes only $O(nm)$ computation time, where nm is the problem size.

As future work, at first, we relax the fair resource allocation condition given by Theorem 1. According to (32), α takes very conservative value when H_{ij}, h_{ij}, and R_j are different largely each other. However, it is shown by simulation experiment that fair resource allocation can be achieved even if (32) does not hold. Moreover, we will extend our method to real-time systems with for multi-dimensional QoS levels. It is also future work to implement the proposed controller in real-time OS.

References

[1] J. Liu, "Real-Time Systems," pp. 394–419, Prentice Hall, 2000.

[2] R. Rajkumar, C. Lee, J. Lehoczky, and D. Siewiorek, "A Resource Allocation Model for QoS Management," Proceedings of the 18th IEEE Real-Time Systems Symposium, pp. 298–307, December 1997.

[3] C. Lee, J. Lehoczky, D. Siewiorek, R. Rajkumar, and J. Hansen, "A Scalable Solution to the Multi-Resource QoS Problem," Proceedings of the 20th IEEE Real-Time Systems Symposium, pp. 315–326, December 1999.

[4] T. Abdelzaher, E. Atkins, and K. Shin, "QoS Negotiation in Real-Time Systems and Its Application to Automated Flight Control," Proceedings of the 3rd IEEE Real-Time Technology and Applications Symposium, pp. 228–238, June 1997.

[5] T. Abdelzaher and C. Lu, "Modeling and Performance Control of Internet Servers," Proceedings of the 39th IEEE Conference on Decision and Control, vol.3, pp. 2234–2239, December 2000.

[6] D. Steere, A. Goel, J. Gruenberg, D. McNamee, C. Pu, and J. Walpole, "A Feedback-Driven Proportional Allocator for Real-Rate Scheduling," Proceedings of the 3rd Symposium on Operating Systems Design and Implementation, January 1999.

[7] C. Lu, J. Stankovic, S. Son, and G. Tao, "Feedback Control Real-Time Scheduling: Framework, Modeling, and Algorithms," Real-Time Systems, Vol.23, No.1–2, pp. 85–126, 2002.

[8] K. Kang, S. Son, J. Stankovic, and T. Abdelzaher, "A QoS-Sensitive Approach for Timeliness and Freshness Guarantees in Real-Time Databases," Proceedings of the 14th Euromicro Conference on Real-Time Systems, pp. 203–212, June 2002.

[9] C. Curescu and S. Nadjm-Tehrani, "Time-Aware Utility-Based QoS Optimisation ," Proceedings of the 15th Euromicro Conference on Real-Time Systems, pp. 83–94, June 2003.

[10] H. Hassan, J. Simo, and A. Crespo, "Enhancing the Flexibility and the Quality of Service of Autonomous Mobile Robotic Applications," Proceedings of the 14th Euromicro Conference on Real-Time Systems, pp. 213–222, June 2002.

[11] F. Harada, T. Ushio, and Y. Nakamoto, "Adaptive Resource Allocation Control with On-Line Search for Fair QoS Level," 10th IEEE Real-Time and Embedded Technology and Applications Symposium, pp. 352–359, May 2004.

[12] G. Buttazzo, *Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications*, Kluwer Academic Publishers, Boston, 1997.

[13] Y. Diao, J. Hellerstein, A. Storm, M. Surendra, S. Lightstone, S. Parekh, and C. Garcia-Arellano, "Incorporating Cost of Control into the Design of a Load Balancing Controller," Proceedings of the 10th IEEE Real-Time and Embedded Technology and Applications Symposium, pp. 376–385, May 2004.

[14] D. McNamee, C. Krasic, K. Li, A. Goel, E. Walthinsen, D. Steere, and J. Walpole, "Control Challenges in Multi-Level Adaptive Video Streaming," Proceedings of the 39th IEEE Conference on Desicion and Control, vol.3, pp. 2228–2233, December 2000.

[15] http://ushiolab.sys.es.osaka-u.ac.jp/~harada/cdc05_fairQoS_050831.pdf