一种改进的自适应多媒体任务调度算法*)

洪雪玉 张 凌 袁 华
(华南理工大学广东省计算机网络重点实验室 广州 510640)

摘 要 支持多媒体任务调度以满足其性能需求,是一项重要而富有挑战性的工作,一直备受关注,并出现了一些实时任务模型。它们都需要任务提供最坏执行时间(Worst Case Execution Time, WCET),以方便准入控制机制的实现,但这正是多媒体任务难以提供的。那么在WCET未知的前提下,如何实现多媒体任务的调度,而且必须支持准入控制和动态QoS控制机制,支持尽可能多任务的执行,使CPU资源的利用最大化?本文首先提出了一种改进的基于速率的自适应(Adaptive Rate-Based, ARB)任务模型。然后通过理论分析和实验证明了:在WCET未知的情况下,基于ARB任务模型的多媒体任务调度算法、准入控制和自适应QoS控制机制是可行的、有效的,而且可以支持尽可能多任务的执行,达到了预期的目标。

关键词 多媒体任务调度,任务模型,自适应QoS,最坏执行时间

An Improved Adaptive Task Scheduling for Multimedia Applications

HONG Xue-Yu ZHANG Ling YUAN Hua
(Guangdong Key Laboratory of Computer Network, South China University of Technology, Guangzhou 510640)

Abstract Much research has been done on guarantee the QoS of multimedia systems, and several real-time task models have been studied. But the Worst Case Execution Time (WCET), which is hard to be provided for multimedia tasks, is needed in these models to make admission controlling possible. Thus, how to effectively schedule multimedia tasks with admission controlling and self-adaptive QoS controlling and more fully resource utilization if WCET is unknown. First, an improved task model, called Adaptive Rate-Based task model (ARB) with WCET unknown, is presented. Then through theoretical analysis and simulation experiment, it is demonstrated to be feasible and effective to schedule multimedia tasks with admission controlling and self-adaptive QoS controlling based on ARB task model.

Keywords Multimedia task scheduling, Task model, Self-adaptive QoS, Worst case execution time

1 引言

随着计算、通信、多媒体处理技术的发展,多媒体应用系统将成为最主要的应用之一。它将包括多种多样的实时任务,如实时控制等硬实时任务、以多媒体为代表的软实时任务,以及信息显示等非实时任务,而多媒体信息将成为最主要的处理对象。在该类型应用系统中,多媒体任务具有下列特征:

·多媒体任务的最坏执行时间(the Worst Case Execution Time, WCET)很难确定。例如执行多媒体压缩解压的任务,每次执行时间可能不一样,甚至变化很大。这种情况在分布式多媒体系统中更为严重,因为不同节点的计算能力不同,所以提供WCET更为困难。

·相关研究表明[1],在实际应用尤其是分布式多媒体系统中,任务事件的触发在任意时刻看来都是随机的,但任务在执行时往往有这样的要求:1)单位时间内执行任务多少次;2)相邻两次执行的开始时间或结束时间必须满足一定的约束。如视频显示任务要求每秒显示30帧,则如果只是简单地保证每33ms显示一帧的话,可能产生抖动现象,因为相邻两帧的显示间隔可能很小也可能很大。因此,除了保证每33ms显示一帧,还应当保证相邻两帧之间的显示间隔约为33ms。

* 与传统实时应用不同的是,多媒体应用需要动态的、灵活的资源分配机制。如不同质量的视频流对于网络带宽、CPU处理带宽、存储空间等资源的需求会有很大的差异。

因此,如何有效地支持多媒体任务的执行以满足其QoS需求,是一项重要而富有挑战性的工作。它需要包括网络、端到端协议、数据库系统和操作系统在内的所有系统部件提供相应的支持。其中,操作系统需要在底层存储、传输和处理器任务调度等各个子系统中提供实时性支持,需要以一种确定性方式实现对CPU、I/O总线、磁盘以及网络带宽等资源的分配,而基于处理器的任务调度是操作系统最基本的研究内容之一。本文将致力于任务调度方面的研究,力图为多媒体任务提供有效的调度支持。

2 相关工作

大多数的实时任务模型[2~9]都是基于Liu和Layland提出的周期任务模型[10],以及Mok提出的零星任务模型[11]。但这两种任务模型均难以准确地描述分布式多媒体系统中软实时任务的特征。例如对于执行视频流压缩解压的任务而言,由于从网络或外部存储设备获取数据所需要的时间是不确定的,因此如果将每次解压并显示一帧图像作为一个作业,那么这种作业的释放没有严格的周期性。但零星任务模型也不能很好地描述这种任务的特征,因为从长期来说,这种任务的执行具有统计意义上的规律性,例如每秒钟处理多少图像

*)基金资助:国家CNGI项目(CNGI-04-15-8A)。洪雪玉 博士研究生,主要研究方向:实时操作系统、多媒体操作系统。

帧。为了解决这个问题，Kevin Jeffay 等人提出了一种介于上述两种任务模型之间的 RBE(Rate-based Execution，基于速率的调度)任务模型[1]。所谓基于速率的调度是指：资源调度程序可以为系统中的软实时任务提供诸如“一个任务的作业以每 y 时间内完成 x 次的平均速率执行”这样的调度支持。这种资源调度方法与实时多媒体应用对于资源的需求有着比较直接的联系，因而具有很高的实用价值。

一个 RBE 任务用一个四元组描述为：$T_i=(x, y, d, e)$。其中，y 表示一段时间间隔；x 表示在 y 时间内 T_i 指定执行的最大次数；d 表示任务从触发到结束被允许的最大时间间隔，即相对截止期限；e 表示任务的 WCET。从本质上来说，RBE 模型为那些具有统计意义上的周期性的多媒体软实时任务确定了一种在 EDF 调度环境下计算作业截止期限的方法。假设一个多媒体软实时任务 T_i 的第 j 个作业 J_{ij} 在 t_{ij} 时刻被释放，且根据应用的要求，该作业必须在 d_{ij} 时间间隔内完成。那么在 EDF 调度中，按照 RBE 任务模型中该任务的作业在 y 时间内执行 x 次的规定，作业 J_{ij} 的绝对截止期限 D_{ij} 被定义为：

$$D_{i,j}=\begin{cases} t_{ij}+d_{ij}, \text{if}(1\leqslant j\leqslant x) \\ \max\{t_{ij}+d_{ij}, y+D_i, j-x\}, \text{if}(j>x) \end{cases} \quad (1^*)$$

这种截止期限的计算方法与直观上理解的每 y 时间内执行 x 次作业的思想相吻合，体现了 RBE 任务模型的两个重要属性：1)可以有一个任务的最多达 x 个连续的具有相同截止期限的作业来竞争处理器；2)作业 J_{ij} 和其前 x 个作业的截止期限最少相差 y 时间。

但是，根据“引言”部分可知，多媒体任务的 WCET 很难确定，而且任务相邻两次执行之间往往有时间上的约束。显然这两点是 RBE 模型没有考虑的。同样，周期任务模型和零星任务模型均需要任务提供 WCET。因此，本文要做的尝试和努力是：在 WCET 未知的前提下，如何实现多媒体任务的调度，而且必须支持下列目标：1)实现准入控制和动态 QoS 控制机制；2)任务相邻的两次执行具有时间约束；3)在不影响系统原有任务顺利执行的前提下，支持尽可能多任务的执行，使 CPU 资源的利用最大化。为此，在 RBE 任务模型的基础上，提出了一种改进的基于速率的自适应(Adaptive Rate-Based，ARB)任务模型，通过 ARB 任务模型的描述、调度算法的设计和实现、实验验证等 3 个方面较完整地说明这种改进的可行性和有效性。

3 ARB 任务模型

定义 $T=\{T_1, T_2, \cdots, T_n\}$ 为一个含有 n 个 ARB 任务的集合。任务 T_i 不断地接收和处理到达的事件，它的每次执行称为一个 job，记为 $J_{ij}(i=1,\cdots,n, j=1,2,3,\cdots)$。任务描述为：$T_i=(X_i, Y_i, C_i, \text{QoS}_i\_\text{lost})$。其中，$Y_i$ 表示一段时间间隔；X_i 表示在 Y_i 时间内 T_i 指定执行的最大次数或平均次数；C_i 表示任意相邻的两个 job 之间，如 J_{ij}，$J_{i,j+1}$，执行时的时间距离约束。具体地，定义为相邻两个 job 的最早可调度时间(Eligible Start Time，EST)的间隔，且 $C_i=Y_i/X_i$；QoS_i_lost 表示 T_i 的执行性能被保证的程度。记 exp_total 和 real_total 分别表示在一定时间间隔内任务期望执行的总次数(或最大次数)和实际被执行的总次数，则 QoS_i_lost 定义为：

$$\text{QoS}_i\_\text{lost}=(\text{exp\_total}-\text{real\_total})/\text{exp\_total}$$

一般地，将二元组 (X_i, Y_i) 称为执行速率[1]，它直观地描述多媒体应用的 QoS 需求，如 Y_i 时间间隔内解压并显示 X_i 个图像帧。

由于任意相邻的两个 job 之间有执行时间上的距离约束，因此 T_i 的执行具有如下时间属性。记 J_{ij} 的到达时刻为 t_{ij}。记 $EST_{i,j}$，$RST_{i,j}$，$RFT_{i,j}$ 和 $D_{i,j}$ 分别为 J_{ij} 的最早可调度时间(Eligible Start Time)、实际开始调度时间(Real Start Time)、实际完成时间(Real Finish Time)和绝对截止期限。它们的定义和关系如下：

$$EST_{i,j}=\begin{cases} t_{ij} & \text{if } j=1 \\ \max\{RST_{i,j-1}+C_i, RFT_{i,j-1,\text{或}j}\} & \text{if } j>1 \end{cases} \quad (2^*)$$

$$D_{i,j}=EST_{i,j}+C_i \quad (3^*)$$

由此可见，T_i 的执行具有下列重要特征：1)当多个事件同时到达时，它们不会同时处于可被调度执行的状态，任意相邻的两个 job 之间的最早可调度时间间隔至少为 C_i。2)只有当前 job 的 $EST_{i,j}$ 和 $D_{i,j}$ 是确定的，下一个 job 的 $EST_{i,j}$ 要等待当前 job 结束之后才能确定，即是动态确定的。3)对于任意 job 而言，必然存在 $RST_{i,j}\geqslant EST_{i,j}$。$RST_{i,j}$ 与 $EST_{i,j}$ 的差值越大，表示系统负载越重，导致任务被调度执行的时间越晚，甚至导致任务的截止期限无法被满足。4)$EST_{i,j}$ 是由 $(RST_{i,j-1}+C_i)$ 决定，而不是由 $(EST_{i,j-1}+C_i)$ 决定的，且 $RST_{i,j}\geqslant EST_{i,j}$，$EST_{i,j}-EST_{i,j-1}\geqslant C_i$，说明了任务具有“滞后”执行的特点，这样可能会导致在一段时间后，任务相邻的两次执行并不处于相邻的两个周期(周期长为 C_i)内，也就是任务实际执行的次数少于期望执行的次数。当系统负载越重，这个问题越突出。参数 QoS_i_lost 正是基于这个问题而定义的，因此它可以反映系统当前负载状态，从而为系统的准入控制和自适应 QoS 控制提供依据。

4 基于 ARB 任务模型的准入控制和自适应 QoS 控制机制

为了保证系统原有任务的顺利执行，当有新任务到达时，需要对其进行准入控制判断，即通常意义上的任务可调度性判定。到目前为止研究的绝大多数任务模型，都是通过计算 cpu 使用率来实现任务可调度性判定，从而实现任务的准入控制功能。但前提是必须提供任务的 WCET。因此，在 ARB 模型中，不能采用计算 cpu 使用率的方法，而是通过 QoS_i_lost 参数实现任务的准入控制，以及动态 QoS 协商(注：文中的 QoS 指的是任务的执行速率)。

基于 ARB 任务模型的准入控制和自适应 QoS 控制的原理如下：当有任务退出时，系统可用资源随之增加。此时，系统将与任务进行 QoS 协商，以期提高其 QoS，最终体现在 QoS_i_lost 值减小了。当有任务加入时，系统必须在保证原有任务 QoS 的前提下，尽可能支持新任务的执行。具体算法如图 1。在新任务参与调度的前提下，新任务集运行 kC_{max}(k 为常量) 时间，然后观察旧任务集中各个任务的 QoS_i_lost 值以及新任务 t_m 的 QoS_i_lost 值是否得到满足。如果旧任务的 QoS_i_lost 不增加，且 t_m 的 QoS_i_lost 值也得到保证，那么就接受 t_m。否则降低新任务的 QoS 请求或与旧任务集中的任务进行 QoS 协商，直到接受或拒绝 t_m。因此，此算法可以实现对新任务准入控制的功能。此外，当系统中现有任务提出新的 QoS 请求或系统可用资源发生变化而导致系统自动向现有任务提出新的 QoS 协商请求时，也是通过该算法实现的。

```
BEGIN   QoS_Adapt (tm)        // (tm,  input parameter
        QoS_T_Set = T - { tm }    // T, task set
L1:     Cmax = max { Ci | 1 ≤ i ≤ n }
        After kCmax interval, doing the following work // k is a constant
        IF every QoSi_lost doesn's increase and QoS of tm is guarantee
                                                  // ti ∈ QoS_T_Set
              ACCEPT request from tm
              RETURN
         ELSE
              IF tm is negociatable
                    negociate with tm to degrade its QoS
                    goto L1
              ELSE
L2:                 IF non_empty (QoS_T_Set)
                          choose tj with the mininum QoSi_lost from QoS_T_Set
                          negociate with task tj to degrade QoS
                          IF succeed   // means that tj agrees to degrade its QoS
                                goto L1
                          ELSE
                                QoS_T_Set = QoS_T_Set - { tj }
                                goto L2
                    ELSE
                                REJECT request from tm
                                RETURN
END
```

图 1　准入控制和自适应 QoS 算法

这里有几个值得讨论的问题：1）当 QoS_i_lost 为 0 时，说明任务可以顺利执行；当 QoS_i_lost 满足不等式：$0<QoS_i\_lost\leqslant\varepsilon$，$\varepsilon$ 足够小且根据应用类型定义，说明任务实际执行次数少于期望执行次数，但结果是可接受的。2）新任务的加入或现有任务提出更高的 QoS 请求时，在接下来一段很短的不被察觉的 $kCmax$ 时间（一般为毫秒级）内，系统执行准入控制判断，因此可能影响其他任务的执行，从而影响了它们的 QoS。这里的解决方法是，将任务分为两种：QoS 不可协商型和 QoS 可协商型。对于前一种任务，将保证它们的 QoS_i_lost 满足：$0<QoS_i\_lost\leqslant\varepsilon$；对于后一种任务，处理上就相对灵活一些，即在系统负载较大时，适当降低其 QoS；在系统负载较小时，也相应的提高其 QoS。然而，该方法并不能彻底地解决问题。因为当系统中可协商型任务较多时，仍然需要较多的时间进行 QoS 协商。3）由于并不需要任务提供 WCET，因此无法在新任务执行之前进行准入控制判断。图 1 所示的准入控制是先允许新任务执行一段很短的不被察觉的 kC_{max} 时间，然后根据系统原有任务的执行是否受到影响来判断新任务是否可接受，或者是否需要进行 QoS 协商。其中，k 的取值不能太小也不能太大。若太小了（如 k 取值为 1），则 QoS_i_lost 的变化不明显；若太大了，则用户会明显地察觉到原有任务受到影响。4）由于任务的 WCET 不可知，因此当任务的执行速率保持不变，而计算由简单变为复杂时，QoS_i_lost 将发生相应的变化，失败率将提高。此时，系统将根据自适应 QoS 控制算法，与应用进行 QoS 协商，以期适当降低其执行速率，从而提高任务执行的成功率。5）当系统处于重载状态时，仍然能够保证 QoS 不可协商型任务的顺利执行。因此，可以将一些比较重要的任务以 QoS 不可协商的方式运行，从而使得系统在重载时仍然能够保证它们的顺利执行。

5　ARB 任务模型的调度算法

5.1　算法描述

设系统的当前时刻为 t。根据第 3 节中任务执行时间的属性分析，在线调度器可采用如下调度算法：当一个新的任务事件到达时，根据式（2*）计算 $EST_{i,j}$，根据式（3*）计算 $D_{i,j}$。从就绪的并且满足不等式 $EST_{i,j}\leqslant t$，$i=1,2,\cdots,n$，$j=1,2,\cdots,m$ 的任务中选择具有最小 $D_{i,j}$ 的任务执行。若有多个任务具有相同的 $D_{i,j}$，则选择具有最小 $EST_{i,j}$ 的任务执行。如果有多个任务具有相同的 $D_{i,j}$ 和 $EST_{i,j}$，则根据 FIFO 原则选择任务。被选中的任务在执行之前更新 $RST_{i,j}$，执行结束时再更新 $RFT_{i,j}$ 值。如果满足下列条件：

$$RST_{i,j}-RFT_{i,j-1}\geqslant C_i$$

并且

$$\lceil RST_{i,j}/C_i\rceil-\lceil RFT_{i,j-1}/C_i\rceil\geqslant 1$$

说明 $J_{i,j}$ 与 $J_{i,j-1}$ 并不是在相邻的两个周期中执行的，中间至少间隔了一个周期，即实际执行次数比期望的执行次数至少少 1。最后更新 QoS_i_lost 值。这里假设任务的初始执行时间为 0。

5.2　算法性能分析

下面通过两个模拟实验来对基于 ARB 任务模型的调度算法和自适应 QoS 控制算法进行性能分析。

模拟实验一：设系统原有任务集 $T=\{t_1,t_2\}$，任务的执行速率分别为（1，2）和（1，4），执行时间均为 1 个时间单位（100 毫秒）。实验结果表明，QoS_1_lost 和 QoS_2_lost 均为 0，说明两个任务可以顺利执行。然后启动执行速率为（1，2）的新任务 t_3，并获得了任务 t_1，t_2，t_3 的 QoS 统计数据，如图 2 所示。其中，x 轴表示任务执行次数；y 轴表示任务执行的成功率，即每 3 * 4（取 $k=3$，$C_{max}=4$）个单位时间间隔内，（1－QoS_i_lost）的统计值。根据图 2 的实验结果可知，新任务 t_3 的加入影响了原有任务 t_1，t_2 的执行。因此，根据图 1 自适应 QoS 算法，系统将与任务 t_3 进行 QoS 协商，并将其执行速率由开始的（1，2）调整为（1，4）。然后，再次获得每隔 3 * 4 个单位时间，任务 t_1，t_2，t_3 的 QoS 统计数据，如图 3 所示。此时，QoS_1_lost 和 QoS_2_lost 均为 0，表示原有任务 t_1，t_2 可以顺利执行；在第一个 3 * 4 时间间隔后，任务 t_3 的 QoS_3_lost 也为 0，因此也可以顺利执行。实验结果表明：1）参数 QoS_i_lost 确实能够反映系统当前的负载情况。首先当新任务 t_3 以（1，2）速率执行时，根据 EDF 经典调度理论，此时系统的 CPU 带宽需求共为 $1/2+1/2+1/4>1$，说明系统已经

超载，图 2 的实验结果也证明了这一点，即，t_1，t_2 的执行均受到影响。然而当 t_3 以(1,4)速率执行时，此时的 CPU 带宽需求共为 $1/2+1/4+1/4 \leqslant 1$，说明系统未超载，同样图 3 的实验结果也证明了这一点，即，t_1，t_2 和 t_3 均可以顺利执行。(2)基于 QoS_ lost 实现的准入控制和自适应 QoS 控制算法是有效的，可以进行动态的 QoS 协商，从而保证各任务的顺利执行。

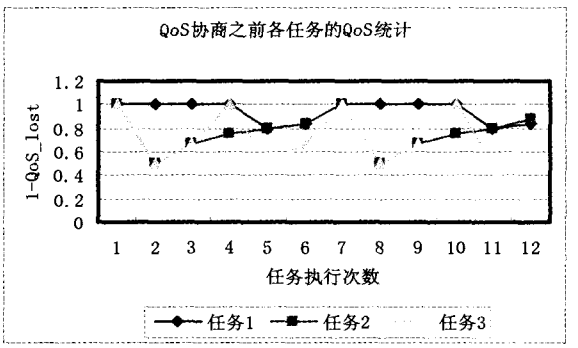

图 2　QoS 协商之前各任务的 QoS 统计

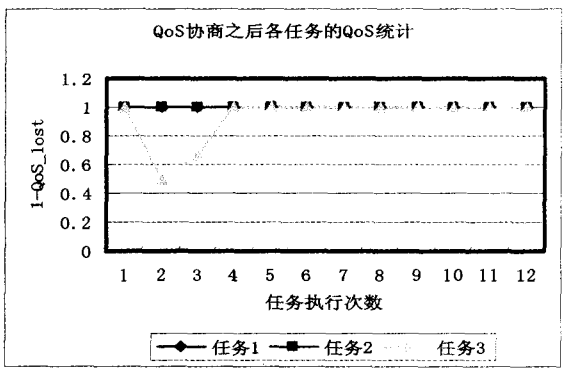

图 3　QoS 协商之后各任务的 QoS 统计

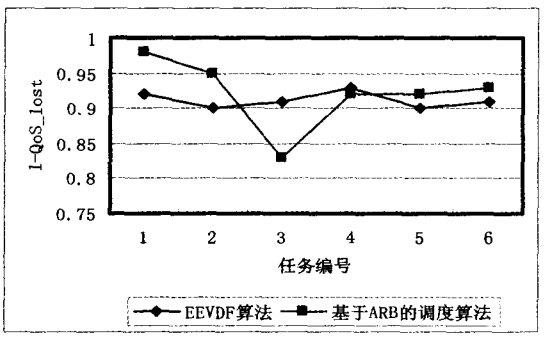

图 4　EEVDF 算法与 ARB 调度算法比较之一

模拟实验二：将 ARB 调度算法与基于比例共享调度算法 EEVDF 进行比较。设有 6 个连续媒体流任务，每个任务的执行速率均为(30,1)，在这里表示任务每秒执行 30 次。此外，每个任务均为 QoS 可协商型任务。实验结果如图 4 所示。其中，当采用 ARB 调度算法时，任务 1 和任务 2 获得了比任务 3 好得多的 QoS 保证。因此，根据自适应 QoS 控制算法，将任务 1 和任务 2 的执行速率分别调整为(20,1)和(25,1)，实验结果如图 5 所示。此时，任务 3 的 QoS 得到了提高。通过实验，可以得出如下结论：1) EEVDF 算法运行更为稳定，使得每个任务的执行结果更为接近，体现了算法资源共享的特征。2)基于 ARB 的调度算法和自适应 QoS 控制算法是有效的，它们在保证系统原有任务执行的同时，可以支持尽可能多任务的执行，以充分利用 CPU 资源。3)采用 EEVDF 算法，应用层 QoS(即执行速率)无法直接映射为内核层的任务权重；而采用基于 ARB 的调度算法，任务 QoS 的改变可以体现为内核层任务的 C_i 值发生变化，从而影响了 $EST_{i,j}$，$D_{i,j}$ 等参数。因此，方便了动态 QoS 协商机制的实现和调度算法的实现；保证了系统过载时，重要任务的优先执行。

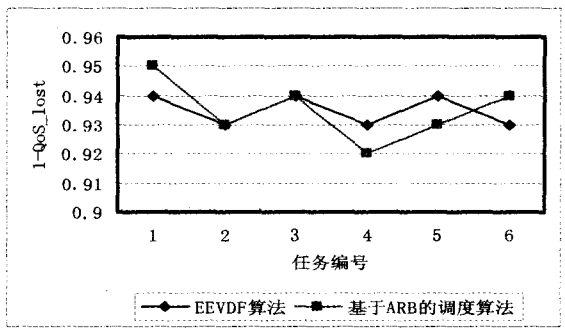

图 5　EEVDF 算法与 ARB 调度算法比较之二

结束语　本文致力于多媒体任务模型和调度控制的研究。在 RBE 任务模型的基础上，提出了一种改进的基于速率的自适应(Adaptive Rate-Based，ARB)任务模型。研究了基于该模型的准入控制和自适应 QoS 控制机制以及调度实现。通过理论分析和模拟实验初步证明了在 WCET 未知的情况下，基于 ARB 任务模型的调度算法、准入控制和自适应 QoS 控制机制是可行有效的。同时，在自适应 QoS 控制机制下，支持了尽可能多任务的支持，从而使系统资源得到更为充分的利用。

参 考 文 献

1　Jeffay K, Goddard S. A Theory of Rate-Based Execution. In: The 20th IEEE Real-Time Systems Symposium, Phoenix, AZ, 1999

2　Jeffay K, Stone D L, Smith F D. Kernel support for live digital audio and video. Computer Communications, 1992, 15(6): 388～95

3　Spuri M, Buttazzo G. Scheduling aperiodic tasks in dynamic priority systems. Journal of Real-Time Systems, 1996, 10(2): 179～210

4　Abeni L, Buttazzo G. Integrating multimedia applications in hard real-time systems. In: The 19th IEEE Real-Time Systems Symposium, Los Alamitos, 1998

5　Palopoli L, Abeni L, Buttazzo G, et al. Real-time control system analysis: an integrated approach. In: The 21th IEEE Real-Time Systems Symposium, Orlando, FL, 2000

6　Kaneko H, Stankovic J A, Sen S, et al. Integrated scheduling of multimedia and hard real-time tasks. In: The 17th IEEE Real-Time Systems Symposium, Los Alamitos, CA, 1996

7　Abeni L, Palopoli L, Buttazzo G. On adaptive control techniques in real-time resource allocation. In: The 12th IEEE Euromicro Conference on Real-Time Systems, Stockholm, 2000

8　Abeni L, Buttasso G. Adaptive bandwidth reservation for multimedia computing. In: The 6th International Conference on Real-Time Computing Systems and Applications, Hong Kong, 1999

9　Jinhwan K, Inhwan J, Bongyour L. Scheduling techniques to integrate MPEG-based multimedia applications in hard real-time systems. In: The Fourth Pacific Rim Conference on Multimedia. Proceedings of the 2003 Joint the Fourth International Conference on Information, Communications and Signal Processing, 2003

10　Liu C L, Layland J W. Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment. Journal of the ACM, 1973, 20(1): 46～61

11　Mok A K-L. Fundamental Design Problems of Distributed Systems for the Hard Real-Time Environment: [Ph D dissertation]. MIT: Dept. of EE and CS, May 1983