

一种帧间相关的窗口限制实时系统的分析与研究

朱响斌,金永贤

(浙江师范大学 信息科学与工程学院,浙江 金华 321004)

(zhu_xb@163.com)

摘要:在研究窗口限制实时系统的基础上提出了一种受窗口限制的帧间相关周期模型。首先对该模型的调度性进行了分析与研究,得出了一些结论,然后提出了相应的动态双窗口限制调度算法,该算法充分利用了帧间相关的特点来提高实时任务的可调度性,减少窗口限制违例。同时,还进行了大量的模拟研究,这些模拟以传统的动态窗口限制调度算法为基准,将其应用于帧间相关的周期模型时的窗口限制违例情况与新算法进行比较,模拟结果表明,新算法的表现优于传统的动态窗口限制调度算法。

关键词:软实时;调度算法;周期任务;实时系统

中图分类号: TP311 **文献标识码:** A

Analysis and research of a window-constrained real-time system with mutual frames

ZHU Xiang-bin, JIN Yong-xian

(School of Information Science and Engineering, Zhejiang Normal University, Jinhua Zhejiang 321004, China)

Abstract: A key problem of real-time operating systems is real-time scheduling policy. Based on the research of Dynamic Window-Constrained Scheduling (DWCS), a periodic window-constrained real-time model with mutual frames was proposed. Firstly, the mode was analyzed and some conclusions about the schedulability of this model were got. In succession, a Dynamic Dual-Window-Constrained Scheduling (DDWCS) was proposed which use the characteristic of mutual frames to improve the schedulability of this model. To evaluate the performance of the new algorithm, extensive simulation studies have been done. These simulations apply DWCS algorithm to schedule the periodic window-constrained real-time model with mutual frames and use it as a baseline to compare with the proposed algorithm. Simulation results show that the new algorithm is better than the old algorithm.

Key words: soft real-time; scheduling algorithms; periodic tasks; real-time systems

近年来随着计算机网络以及多媒体的技术的发展,实时系统的发展呈现出许多新的特点。比较突出的有视频点播、视频会议、数字图书馆以及远程教学等这样的软实时系统越来越普及。这些软实时系统往往要求在特定的时间内处理完特定数量的流媒体信息,并且要求端系统和网络系统都必须支持 QoS,因而这些软实时系统就需要有一种能很好地满足其 QoS 要求的调度模型。该调度模型应具有相当大的灵活性,这种灵活性体现在能够定量的描述软实时系统满足时限与不满足时限的情况,譬如视频播放或音频播放,系统由于过负载等原因在一段时间内损失 10 个帧(即不满足时限),但这 10 个帧是连续损失的还是间断损失的其 QoS 显然是不一样的,这就要求调度算法在该系统损失帧时能很好地控制损失帧情况,使系统能较好地满足其 QoS 的要求。

为了解决这个问题,就需要对传统的周期模型^[1]进行扩展,对此现在主要提出了 (m, k) 模型^[2]和窗口限制模型^[3,4]。 (m, k) 模型要求在 k 个连续的帧中至少要满足 m 个帧,这样通过 m 与 k 两个值对实时系统的时限满足情况进行了定量的描述。在 (m, k) 模型的基础上,文献[5]又提出了弱硬实时系统概念,弱硬实时系统强调了不满足时限的情况是受严格限制的。窗口限制模型与 (m, k) 模型类似,但 (m, k) 模型中 k 个

连续的帧组成的窗口在 (m, k) 中是重叠的,而在窗口模型中是不重叠的。不幸的是这两个系统在一般情况下的可调度性都是强 NP 难题^[6],但是由于在窗口限制系统中, k 个连续的帧组成的窗口是不重叠的,因而在一定的条件下可以验证其可调度性^[3,4]。因此窗口限制实时系统能够很好地满足新一代实时系统要求,特别适合于流媒体和网络数据传输中的服务质量控制和任务调度。

但是在实际应用中,实时任务在其每个请求之间往往存在相关性,也就是说必须相关的几个请求都完成时限才有意义,譬如 MPEG 中的 I, B, P 帧,它们之间是相关的。又如在工业数据传输中以及网络通信中,一个数据段分成几个数据包传送,若其中一个数据包丢失了,则相当于整个数据段全丢失。又如在大规模传感器网络中,一个有效数据很可能是许多传感器组传送的数据帧组成的。基于上述情况,我们提出了一种帧间相关的实时周期模型。

帧间相关的周期模型不同于周期多帧任务模型^[7,8],周期多帧任务模型也是对传统周期模型的一种扩展,但它是针对任务中不同帧具有不同的执行时间而建立的,譬如 MPEG 中的 I, B, P 帧的执行时间是不相同的,其帧的执行时间不再是一个单一的最大执行时间,而是由几个执行时间值组成的

一个执行时间向量。而帧间相关的周期模型考虑的是帧间相关的特点,譬如 MPEG 中的 I, B, P 帧之间的相关性。

由于新的周期模型所具有帧间相关的特点,因此如果仍然采用动态窗口限制调度(DWCS)算法^[3,4],就很有可能浪费系统资源,如带宽或 CPU 时间,因为在某帧已不满足时限情况下,DWCS 算法仍然让与其相关的后续帧参与调度。因此本文在动态窗口限制调度算法的基础之上,提出了一种新的动态调度算法—动态双窗口限制调度(DDWCS)算法,该算法充分利用了新的周期模型所具有的帧间相关的特点,来提高模型可调度性和服务质量。

1 周期任务模型

在传统的周期模型的调度中,往往假定在一个任务中,每个请求满足时限与否是相互独立的,也就是说一个请求不满足时限不会影响到下一个请求满足时限的意义。在本文中我们认为一个任务中的各个请求是存在某种关系,也就是说某一个请求满足时限是否有意义,还要看与其相关的请求是否已满足时限,如果与其相关的请求没有满足时限,则其满足时限也没有意义。本文假设任何时候优先级高的任务可抢占低优先级任务。

1.1 窗口限制的帧间相关周期任务模型

我们把周期任务的请求(invocation)称为帧。则有如下定义:

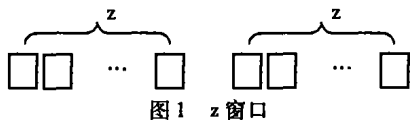


图1 z窗口

定义1 满足 z 时限指的是在帧间相关的周期性任务中,在一个连续 z 个帧组成的固定窗口内满足该窗口内所有的时限;不满足 z 时限指的是在帧间相关的周期性任务中,在一个连续 z 个帧组成的固定窗口内不满足其中任意一个时限。

定义2 一个帧间相关的周期性任务是一个由 N 个任务 $\tau_i (1 \leq i \leq N)$ 构成的集合 Π , 其中 τ_i 是一个由四元组 (T_i, C_i, W_i, z_i) 所确定的任务 $(C_i \leq T_i)$ 。其中:

T_i : 任务 τ_i 的周期。

C_i : 任务 τ_i 的计算时间。

z_i : 任务 τ_i 的固定窗口大小。在该窗口内具有连续 z_i 个请求帧,窗口内的请求帧是相关的,也就是说,该窗口内的请求帧的时限都满足时限才有意义,否则等同于窗口内所有请求帧都不满足,固定窗口如图1所示。

W_i : 任务 τ_i 的窗口限制,其值又等于 x_i/y_i , 其中 x_i 为在每个固定窗口 y_i 中可以不满 z 时限的数值, y_i 为固定窗口的大小。因此对于每个任务来说,在其固定窗口 y_i 之内必须满足 $y_i - x_i$ 个的 z 时限。否则就称为产生窗口限制违例(window-constrained violation)。

该周期模型各任务时限依据软实时特点,其值分别为各任务的周期 T 。譬如一个任务为 $(5, 2, 2/3, 2)$, 其中 $z = 2$, 这样第 i 与 $i+1 (i = 0, 2, 4, \dots)$ 构成一个 z 窗口,也可以看成是一个由两个帧组成的 z 帧,在这个 z 窗口的基础之上,增加 W 窗口限制,也就是可以在连续的三个 z 窗口中不满足其中两个 z 时限。如果没有满足窗口限制,也就是某三个 z 窗口组成的 y 窗口中不满足其中三个 z 时限,则称为产生窗口限制违例。软实时调度的目的是少产生窗口限制违例,尽量满足以窗口限制为基础的 QoS 要求。

1.2 利用率

利用率如果是针对网络通信,则可以指带宽利用率,如果是针对处理机来说,则可以指处理机利用率。在本文中指的是

处理机的利用率。如前所述,对于 W_i 为 x_i/y_i 的任务 τ_i 来说,最少必须有 $y_i - x_i$ 个 z 固定窗口在任务 τ_i 中的每个 y_i 窗口中满足其 z 时限。因此根据文献五对利用率的定义,对于任务 τ_i , 其最小的处理机利用率 $U_i = \frac{(y_i - x_i)z_i C_i}{y_i T_i z_i} = \frac{(y_i - x_i)C_i}{y_i T_i}$ 。因而对具有 n 个任务的实时任务集来说,其最小的利用率为 $U = \sum_{i=1}^n \frac{(1 - W_i)C_i}{T_i}$ 。但利用率大于 1 的时候,必然会产生窗口限制违例。

2 动态双窗口限制调度算法(DDWCS)

2.1 算法思想

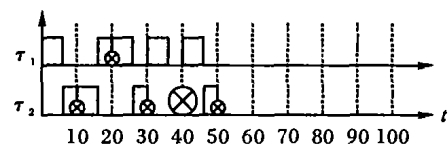
文献[3]提出了动态窗口限制调度算法(DWCS), DWCS 算法的思想是依据当前的窗口限制与时限来确定优先级。其优先级的确定是与 EDF 算法相结合的,但我们主要关心的是窗口限制部分是如何影响优先级变化的。DWCS 算法的核心是当前窗口限制 W 是动态变化的,而 W 的变化是与前面帧的满足时限情况是密切相关的。

但是在帧间相关的周期任务的情况下,基于 DWCS 方法显然需要改进,因为它没有把帧间相关的特点考虑进去。譬如考虑表 1 所示的任务集 1 ($U > 1$):

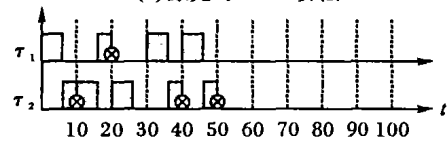
表1 任务集1

任务	τ_1	τ_2
T	10	10
C	6	6
z	3	1
W	1/3	1/2

如果采用原先的 DWCS 的算法,不考虑帧间相关的特性,则调度产生窗口限制违例,如图 2(a) 所示,而采用图 2(b) 调度则可以少产生窗口限制违例,因而我们需要考虑帧间相关的特性。由于对于 z 大于 1 的情况,如果其中一帧失败,则后续帧的调度已经没有任何意义,因此可以考虑在任务集中设一最低的优先级,只要有其他任务需要调度则其必被抢占。对于上述 DWCS 有如下的改进,一旦 z 窗口内不是最后一帧调度失败,则建立一个标记,使该 z 窗口内其余帧不再参加调度。这样我们在确定下一个请求的优先级时,不但要考虑窗口限制的满足情况,同时还要考虑 z 窗口内的情况。



(a) 原先的DWCS算法



(b) 采用改进后的DWCS算法

图2 采用原先的 DWCS 算法产生窗口限制违例

基于上述两种情况,可以设置两个窗口:一是原先的窗口限制 W , 另一是在 z 窗口内的限制情况,我们把 z 窗口认为是一个 W 为 0 的窗口限制情况。综合这两个限制情况再计算出其下一个请求的优先级,我们把这种方法称为动态双窗口限制调度算法(DDWCS)。

2.2 算法设计

该算法的设计是基于原先的 DWCS 算法的。由于在 z_i 窗口之内其帧间是相关的,因此其中任一帧失败,则 z_i 窗口之外的其他帧无须参与调度。这可以增加一个 $ztag_i$ 标志来解决,该标志用来记录在任务 τ_i 的 z_i 窗口之内的帧的满足时限的情况。

首先设任务 τ_i 原始的窗口限制是 $W_i = x_i/y_i$, 帧间相关的窗口大小为 z_i , $W_i' = x_i'/y_i'$ 是当前的窗口限制值, z_i' 是当前

的 z 窗口大小值。在初始情况下, $W_i = W_i', z_i = z_i'$ 。当上一帧在其时限前完成任务时, W_i' 与 z_i' 的调整算法如下:

算法 1:

```
if ( $z_i' = 1$ ) then
   $z_i' = z_i$ ;
   $ztag_i = 0$ ;
  if ( $y_i' > x_i'$ ) then  $y_i' = y_i' - 1$ ;
  else if ( $y_i' = x_i'$ ) and ( $x_i' > 0$ ) then
     $x_i' = x_i' - 1$ ;  $y_i' = y_i' - 1$ ;
  if ( $x_i' = y_i' = 0$ ) or ( $\tau_i$  is tagged) then
     $x_i' = x_i$ ;  $y_i' = y_i$ ;
  if ( $\tau_i$  is tagged) then reset tag;
else  $z_i' = z_i' - 1$ ;
```

同时,对于不满足时限的任务 τ_j 的 $W_j'(j \neq i)$ 与 z_j' 的调整如下:

算法 2:

```
if ( $z_i' = 1$ ) then
   $z_i' = z_i$ ;
   $ztag_i = 0$ ;
  if ( $x_j' > 0$ ) then
     $x_j' = x_j' - 1$ ;  $y_j' = y_j' - 1$ ;
  if ( $x_j' = y_j' = 0$ ) then
     $x_j' = x_j$ ;  $y_j' = y_j$ ;
  else if ( $x_j' = 0$ ) and ( $y_j > 0$ ) then
    Tagged  $\tau_i$  with a violation;
else
   $z_i' = z_i' - 1$ ;
   $ztag_i = 1$ ;
```

优先级排列改动如下:

表 2 优先级顺序

优先级排列(从高到低)
时限小的且 $ztag$ 为 0 的优先级为高
相同时限,则 W 值低且 $ztag$ 为 0 的优先级为高
相同时限且 W 值为 0,则 y 值大且 $ztag$ 为 0 的优先级为高
相同时限且 W 值相等(不为 0),则 x 值小且 $ztag$ 为 0 的优先级为高
其他 $ztag$ 为 0 的情况按照 $FCFS$ 排列

调度算法如下:

While(TRUE)

```
  根据表 2 从各任务中的每个待调度任务中找出优先级最高的任务  $\tau_i$ ;  
  任务  $\tau_i$  的当前帧得到服务;  
  根据算法 1 调整  $W_i'$ ;  
  调整当前任务的时限;  
  对于每个其他的任务  $\tau_j(j \neq i)$ ,如果其产生不满足时限情况,则根据算法 2 对  $W_j'$  进行调整同时调整其时限。
```

2.3 算法复杂度

DDWCS 的复杂度分为两部分:1)根据表 3 选择下一个任务的时间;2)调整当前窗口限制值与帧的时限所需要的时间。其中第一项的时间复杂度可以达到 $O(\log(n))$, n 为任务数。但是第二种情况,当所有的非当前任务都不满足时限时,第二项的时间复杂度达到 $O(n)$ 。因此总的复杂度为 $O(n)$,这是与 DWCS 算法的复杂度是相同的。

3 模拟研究与分析

为了对新算法的性能进行评估,进行了广泛的模拟研究。

主要对比两种算法在新的周期模型下的总窗口违例数与利用率 U 、 z 窗口大小以及 W 的关系。

我们把周期模型的 T_i, W_i, z_i 等参数限制在一定的范围内,在带宽利用率 U 的限制下,随机生成周期任务集,每次生成 2000 个周期任务集。具体参数如表 3 所示,每个实验值取 5 次实验结果的均值。在实验模拟中,调度粒度与帧的计算时间是相等的,因此这两个值都取为 1。将调度长度设置为 $1000 + 1000 * Max_z/2$ 。任务的时限都等同于相应任务的周期值。

表 3 模拟参数表

参数	含义	实验中的取值
Max_C	帧的最大计算时间	1
Min_C	帧的最小计算时间	1
$Slot$	调度粒度	1
Max_T	任务的最大周期	8
Min_T	任务的最小周期	2
Max_z	任务的最大 z 窗口值	3
Min_z	任务的最小 z 窗口值	1
Max_y	任务的最大窗口值	10
Min_y	任务的最小窗口值	1
Max_W	任务的最大 $W(x/y)$ 值	0.8
Min_W	任务的最小 $W(x/y)$ 值	0.0

1) 利用率 U 对窗口违例数的影响

实验在 $Max_z = 3$ 以及 $Max_W = 0.8$ 的情况下模拟出利用率与总窗口违例数的变化关系,如图 3 所示。从图 3 可看出,在 $U < 1$ 的情况下,两者的表现类似,但是在 $U \geq 1$ 的情况随着 U 的增加,DDWCS 算法的窗口违例数明显要高于 DDWCS 算法。这说明新的算法在充分考虑了帧间相关的情况下,在系统的过度负载的时候有着比 DWCS 算法非常明显的优势。这是由于在过度负载的情况下,不满足 z 时限的情况急剧增加,而使 DDWCS 算法放弃更多的不需要参与调度的帧,从而节约出带宽满足其他任务的窗口限制。这样就使窗口违例数随 U 增加的速度要比 DWCS 算法情况下的增加速度来得慢。

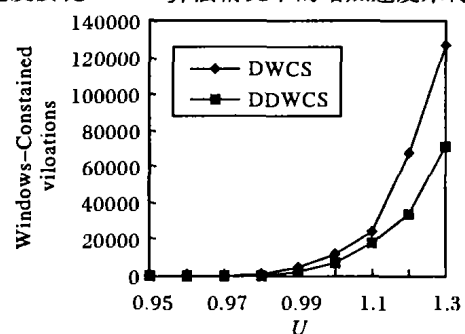


图 3 利用率 U 的影响

2) Max_z 对窗口违例数的影响

实验在利用率 $U = 1.1$ 以及 $Max_W = 0.8$ 情况下模拟出 Max_z 与总窗口违例数的变化关系,如图 4 所示。从模拟结果可看出,在 Max_z 为 1 时,由于此时周期模型已退化为原先的周期模型,DDWCS 算法也退化为 DWCS 算法。因此两者的实验结果是一致的,如图 4 所示。随着 Max_z 的增大,两者的差距就呈现出来了,这表明 DDWCS 算法有着比 DWCS 算法更加好的调度结果。这也就说明了新的算法在充分考虑了帧间相关的情况下,在系统过度负载的时候有着比 DWCS 算法非常明显的优势。而且在 z 窗口增大时,两者的差距越发明显。这是因为一方面由于不满足 z 时限的情况增加,另一方面由于通过新算法节约的带宽也越多,造成新算法的违例数在

(下转第 1814 页)

因此,本文提出的基于混合特征的帧内预测模式决策算法对原算法的优化效果非常显著。

4 结语

针对 H. 264/AVC 参考代码中帧内预测模式决策算法运算复杂度较大的弊端,提出了一种基于混合特征的快速帧内预测模式决策算法。该方法强调了 DC 模式的重要性,并结合混合代价函数特性预先对候选模式进行筛选,由此加速了帧内预测模式决策中最耗时的 4×4 亮度块模式决策环节。通过对测试图像序列的性能参数和 R-D 曲线的分析表明,该方法在保证图像保真度和码率性能的前提下,将 1 帧编码速度提高了近 30%,对原算法具有较显著的优化效果,进而提高了编码系统的效率。为了提高整个系统的效率,今后还可以将本算法应用于解码系统,以进一步增加本算法的实用性。

参考文献:

- [1] Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H. 264 | ISO/IEC 14496-10 AVC) JVT Doc. JVT-G050 [Z]. 2003.
- [2] THOMAS W, GARY JS. Overview of the H. 264/AVC Video Coding Standard [J]. IEEE Transactions on Circuits and Systems for Video

Technology, 2003, 13(7): 560 - 576.

- [3] PENG Y, HYE Y CT, ALEXIS MT, et al. Fast Mode Decision and Motion Estimation for JVT/H. 264 [A]. IEEE International Conference on Multimedia&Expo [C]. Maryland, 2003, III: 853 - 856.
- [4] LIM KP, WU S, WU DJ, et al. Fast INTER Mode Selection [A]. JVT 1020, 9th JVT meeting [C]. 2003.
- [5] RICHARDSON I EG. H. 264/MPEG-4 Part 10 White Paper [S]. 2003.
- [6] Joint Video Team (JVT), reference software JM7.6 [S].
- [7] PAN F, LIN X, SUSANTO R, et al. Fast Mode Decision for Intra Prediction [A]. JVT G013, 7th JVT meeting [C]. 2003.
- [8] MENG BJ, OSCAR CA, WOMG CW, et al. Efficient Intra-Prediction Algorithm in H. 264 [A]. IEEE International Conference on Multimedia&Expo [C]. Maryland, 2003, III: 837 - 840.
- [9] MENG BJ, OSCAR CA. Fast Intra-Prediction Model Selection for 4×4 Blocks in H. 264 [A]. ICASSP 2003 [C]. 2003. 389 - 392.
- [10] MENG BJ, OSCAR CA, WOMG CW, et al. Efficient Intra-Prediction Mode Selection for 4×4 Blocks in H. 264 [A]. ICME 2003 [C]. 2003. 521 - 524.
- [11] KIM CS, SHIH HH, KUO JC-C. Fast Intra-Prediction Mode Selection for H. 264 Codec [A]. SPIE International Symposium ITCOM 2003 [C]. Orlando, Florida, 2003.

(上接第 1782 页)

$Max_z > 2$ 时明显下降,因而在 z 窗口增大时,两者的差距更加明显。

3) Max_W 对窗口违例数的影响

实验在 $Max_z = 3$ 以及 $U = 1.1$ 的情况下模拟出 Max_W 与总窗口违例数的变化关系,如图 5 所示。从图 5 可看出,总体上 DWCS 算法的窗口违例数明显要高于 DDWCS 算法。而随着 Max_W 的增加, DWCS 算法下的窗口违例数的下降不明显,而 DDWCS 算法的窗口违例数却在 $Max_W > 0.3$ 时呈明显的下降趋势,从而使 DDWCS 算法的优势更加明显。这说明新的算法对 W 的变化更为敏感。这是因为 DDWCS 算法在 W 限制放宽的情况下,能够更加容易地利用节约的带宽满足任务的窗口限制,这也就说明了 DDWCS 算法比 DWCS 算法有着更强的调度能力。

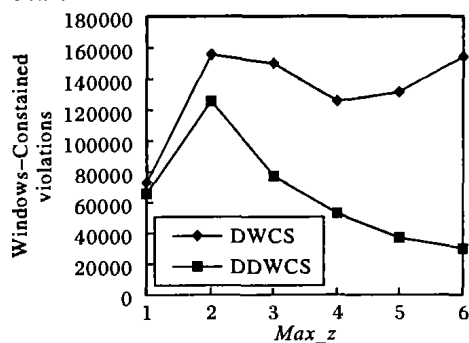


图 4 Max_z 的影响

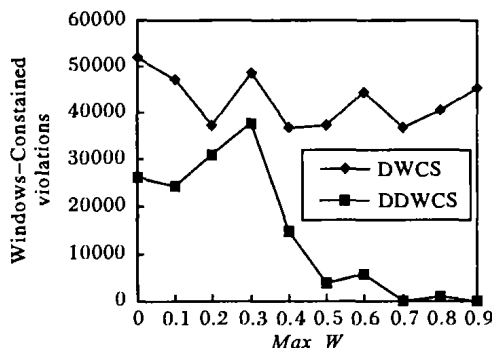


图 5 Max_W 的影响

4 结语

本文提出了一种新的受窗口限制的帧间相关周期模型,同时针对该模型的帧间相关的特点,提出了基于动态双窗口限制调度算法,实验结果表明新算法比旧算法有更好的调度能力,具有非常明显的优势。利用该模型人们可以在帧间相关的实时应用系统中,采用动态双窗口限制调度算法来对 QoS 进行控制以及进行任务调度。

同时随着计算机以及网络多媒体技术的迅速发展,对实时操作系统的发展也提出了新的要求,譬如需增加对网络功能的支持以及相应的服务质量的支持,因此研究支持 QoS 的实时操作系统是当前一个研究热点。本文所研究的调度模型及算法也完全可应用于支持 QoS 的实时操作系统中。

参考文献:

- [1] LIU CL, LAYLAND JW. Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment [J]. Journal of the ACM, 1973, 20(1): 40 - 61.
- [2] HAMDAOUI M, RAMANATHAN P. Evaluating Dynamic Failure Probability for Streams with (m, k)-Firm Deadlines [J]. IEEE Transactions on Computers, 1997, 45(12): 1325 - 1357.
- [3] WEST R, POELLABAUER C. Analysis of a Window - Constrained Scheduler for Real-Time and Best-Effort Packet Streams [A]. Proceedings of the 21st IEEE Real-time Systems Symposium (RTSS) [C]. Orlando: IEEE, 2000. 239 - 248.
- [4] WEST R, SCHWAN K. Dynamic Window-Constrained Scheduling for Multimedia Applications [A]. 6th Proceedings of the IEEE International Conference on Multimedia Computing and Systems (ICMCS) [C]. Florence, Italy: IEEE, 1999. 87 - 91.
- [5] BERNAT G, BURNS A, LIAMOSI A. Weakly Hard Real-Time Systems [J]. IEEE Transactions on Computers, 2001, 50(4): 308 - 321.
- [6] MOK AK, WANG W. Window-Constrained Real-Time Periodic Task Scheduling [A]. In Proceedings of the 22nd IEEE Real-Time Systems Symposium [C]. London: IEEE, 2001. 15 - 24.
- [7] MOK AK, CHEN D. A Multiframe Model for Real-Time Tasks [J]. IEEE Transactions on Software Engineering, 1997, 23(10): 635 - 645.
- [8] 黄文广. 一个改进的实时任务模型—周期多帧任务模型 [J]. 计算机研究与发展, 2001, 38(2): 234 - 239.