文章编号：1000－5471(2007)06－0094－04

现场硬实时网络调度算法研究①

胡 勇[1]， 叶 明[2]

1. 重庆交通大学 应用技术学院，重庆 400042；2. 西南大学 计算机科学与信息工程学院，重庆 400715

摘要：研究了实时任务死线不大于其周期的任务集调度条件与硬实时网络调度算法约束条件，论证了计算时间复杂度，并将单调死线调度方法直接用于非周期任务调度. 最后对调度算法的可调度性进行了严格测试.

关 键 词：实时系统；网络调度；单调死线；调度时间；调度加速比

中图分类号：TP393.02 **文献标识码**：A

硬实时网络调度算法在实时系统中有着广泛的应用，按消息到达模式来分，这类算法包括周期消息调度算法和非周期消息调度算法两种. 周期消息调度算法主要包括单调比率(monontic rate，RM)算法，截止期最早优先(earlist deadline first，EDF)算法，基于距离约束的风车调度(DCTS SR)算法，价值密度最大优先(highest value density first)算法. 在周期调度算法中，RM 被证明是最优的静态可抢占实时调度算法. 实时系统使用 RM 调度算法，任务的优先级按其周期 T 来分配，周期越小其优先级越高. 由于 RM 的优先级由消息周期决定，所以计算优先级开销较低，实施起来容易. 而 EDF 算法是一种动态调度算法，其任务的优先级根据任务的死线动态变化，死线越短，任务优先级越高. 由于 EDF 是一种优先级动态变化的，它能够获得比 RM 更高的处理器利用率，但是 EDF 的缺点是在每一调度时刻都要计算每个消息的死线，并且根据计算结果动态改变任务消息优先级. 距离约束是指，同一消息相邻两次调度完成的时间不大于某一段时间，风车调度算法是把任务集距离约束按照一种规则转化为任务周期，转化后 SR 就按照类似于 RM 调度算法对任务集调度. 价值密度最大优先算法优先调度价值密度最高的任务，它是一种贪婪算法，总是假定具有较高密度和较小空闲时间的任务很快到达，因此执行具有最高价值密度的任务，往往导致许多任务错失不必要的截止期. 由于单调死线实际应用面广，故本文重点讨论单调死线调度理论.

1 硬实时网络调度算法

假定调度任务满足关系：

$$\text{Computation time} \leqslant \text{Deadline} \leqslant \text{Period}(T_i)$$

对于每一个任务 τ_i(τ_1 有最高优先级，τ_i 具有最低优先级)：

$$C_i \leqslant D_i \leqslant T_i$$

单调死线任务优先级顺序在概念上同单调比率优先级顺序相似. 任务所分配优先级高低同死线长度成反比. 因而，具有最短死线的任务被分配给最高优先级，最长死线任务分配给最低优先级. 当任务周期等于其死线时，缺省任务优先级顺序同单调比率顺序一样.

单调死线优先级分配是一种最优静态优先级安排，也即是说，如果任何静态优先级调度算法能够调度其死线不同于周期的任务集，用单调死线任务优先级顺序调度算法也能调度那个任务集. 当然，其时间特点适合单调比率分析的任务集也能被任务死线不同于其周期的静态优先级调度理论认可.

通常，单调死线调度安排没有被采用，主要是因为适当调度算法的缺乏. 单调比率调度算法能够通过

① 收稿日期：2007－04－03
作者简介：胡 勇(1968－)，男，四川仁寿人，讲师，主要从事计算机控制与通信.

减小独立任务周期直到周期逼近其死线的方法而被采用. 当处理器上任务没有被充分估计负载过重时，这种方法不是优化的调度方法.

下面探讨一种对原来可调度性测试方法进行了优化的测试方法. 这种方法建立于关键时间点概念之上，而关键时间点是指所有任务同时释放的时间点. 当此类事件发生时，我们有最坏情况处理器请求. 如果所有任务在关键时刻开始能够满足任务死线，那么它们总是满足任务集任务死线. 基于此建立可调度性测试基础：假定所有任务同时释放，检查单个任务完成的所有任务的完成情况. 一种可调度性测试条件如下：

$$\forall_i : 1 \leqslant i \leqslant n : \frac{C_i}{D_i} + \frac{I_i}{D_i} \leqslant 1 \tag{1}$$

$$I_i = \sum_{j=1}^{i-1} \left\lceil \frac{D_i}{T_j} \right\rceil C_i$$

其中 I_i 是带有冲突任务 τ_i 执行的较高优先级量化参数.

此测试条件成立则任务 τ_i 是可调度的，它的执行时间之和及被高优先级任务执行所带来的冲突一定不大于任务死线 D_i. 在以上条件中，冲突由任务 τ_i 死线到来之前被释放的所有高优先级任务计算产生. 这种测试条件是必要非充分的，当冲突被计算时，在 D_i 之前，I_i 可能比任务 τ_i 遇到的实际冲突大. 给出一种更精确的测试条件如下：

$$\forall_i : 1 \leqslant i \leqslant n : \frac{C_i}{D_i} + \frac{I_i}{D_i} \leqslant 1 \tag{2}$$

在这里

$$I_i = \sum_{j=1}^{i-1} \left\lceil \left\lceil \frac{D_i}{T_j} \right\rceil C_j + \min\left\{ C_j, D_i - \left\lceil \frac{D_i}{T_j} \right\rceil T_j \right\} \right\rceil$$

上面的调度条件成立部分地弥补了高优先级任务不能在 D_i 之前发生、甚至不能在 D_i 之前释放的缺陷.

式(1)和式(2)列出的调度条件约束是充分不必要条件. 为了形成一个充分必要调度条件，调度必须被仿真以至知道高优先级任务精确的插入时间.

如果任务 τ_i 在 t_i(t_i 属于[0，D_i])满足其死线，或可以减少约束方程个数，则不需要在[t_i，D_i]之间仿真方程，并认为在时间[0，D_i]中，任务 τ_i 满足其死线[1]. 同时，由于高优先级任务带来的冲突在这个时间间隔内是单调增加的，当有高优先级任务释放时，冲突增加的时间点发生了. 这可以由图 1 证明.

在图 1 中比 τ_4 优先级高的任务有 3 个，当高优先级任务被释放时，关于 t 的 I_4 单调增加[2]，且没有高优先级任务释放的时间间歇中发生的任务状态转换[3]. 显而易见，当冲突没有改变时对每一个状态仅仅需要一个方程仿真. 为了增大任务执行的有效时间，选择最右边状态点仿真一种可能减少仿真调度方程数量的方法是测试高优先级任务在时间[0，D_i]内释放的所有任务 τ_i 点. 因为一个方程把一个任务集测试视为是可调度的，而不必进一步证明其它方程. 因此，仅仿真在时间[0，t_i]内的方程.

注意到如果[0，D_i]内每一个时间点在 D_i 满足其死线的高优先级任务 τ_i 被释放约束方程数没有发生减少. 但是，可以通过计算任务时间来减少约束方程数量[4].

如图 2 所示，任务集总共计算时间(C_s)要求被计划分配[5]. 在第一个计算时间等于其消逝时间的时间点，我们发现时间点 t_4. 在上图中 t_4 这时间与任务 τ_4 的死线是一致的.

分析图 2 在[0，C_i]中任务 τ_i 可调度性有一个缺点. 也就是，因为在 0 时刻相当于关键时刻(所有任务同时释放时刻)，所以，任务 τ_i 可能被完成的第一个时间点是：

$$t_0 = \sum_{j=1}^{i} C_j$$

这就给出一种调度性约束条件：

$$\frac{I_i^{t_0}}{t_0} + \frac{c_i}{t_0} \leqslant 1$$

在这里：

$$I_y^x = \sum_{z=1}^{y-1} \left\lceil \frac{x}{T_z} \right\rceil C_z$$

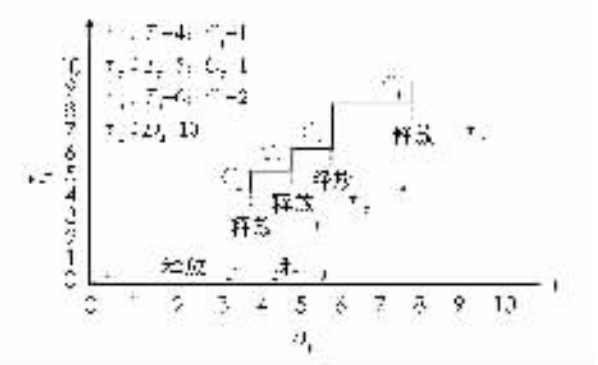

图 1 冲突增加

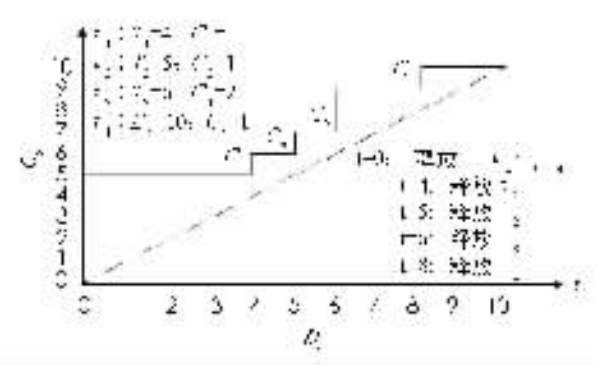

图 2 处理计算时间

假定在[0，t_0]内 t_i 时刻每一个任务释放仅仅一次，如果在时间间歇[0，t_0]内有高优先级任务已经被释放，那么调度将失败[6].

当 t_i 大于其死线时，算法终止，任务 τ_i 同时不能调度. 我们也注意到在任务死线不大于其周期的地方，该算法也能被用来估计任何固定优先级任务集的可调度性，而且这种固定优先级无论采用何种分配规则都满足. 算法实现如下：

$$
\begin{gathered}
foreach\ \tau_i\ do \\
t = \sum_{j=1}^{i} c_j \\
continue = TRUE \\
WHILE[CONTINUE]DO \\
IF\left[\frac{I_i^t}{2} + \frac{C_i}{t} \leqslant 1\right] \\
CONTINUE = FLASE \\
/ * \tau_i\ is\ scheduable\ * / \\
else \quad t = I_i^t + C_i \\
endif \\
if\ [t < D_i] \\
/ * \tau_i\ is\ unsheduable\ * / \\
endif \\
endwhile \\
end for
\end{gathered}
$$

2 可调度性测试

通过测试，该硬实时调度算法的可调度性判定可以用如图 3 的曲线进行描述. 该图说明消息集利用率越高，网络消息集利用率越低，即实时任务集可调度性概率越高；反之，消息集利用率越低，那么网络消息集利用率越高，即实时任务集的可调度性概率越低[7].

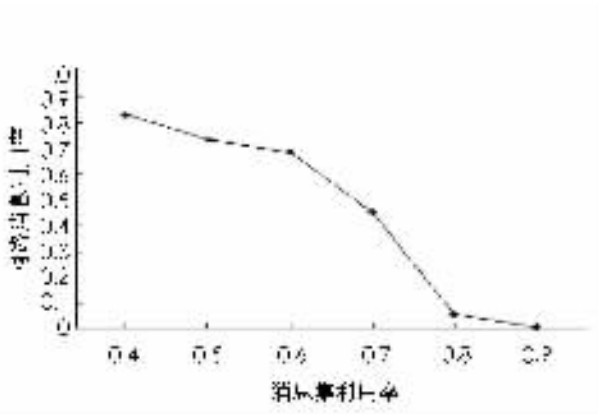

图 3 调度性测试图

3 结 论

研究了任务在什么情况下是可调度的，并用一个简单方程对每一个任务的可调度性进行判断，以此确定了充分必要的可调度性条件.

参考文献：

[1] Tindell K W, Burns A, Wellings J. Allocating hard real-time tasks: An NP-Hard problem made easy [J]. Real-Time Systems, 1990, 4(2): 145－165.

[2] Sha JP L, Ding Y. The Rate-Monotonic Scheduling Algorithm: Exa Characterization and Average Case Behavior [J]. IEEE Real－Time System , 1989, 42(4): 166－171.

[3] Fixed Priority Scheduling of Periodic Task Sets with Arbitrary Deadlines [A]. Proc IEEE Real-Time System Symp [C]. Oakland: IEEE Computer Press, 1990: 201－210.

[4] Baruah S K, Mok A K, Rosier L E. Preemptively Scheduling Hard-Real-Time Sporadic tasks on One Processor [J]. IEEE Real-Time System, 1990, 32(3): 182－190.

[5] 王志平，熊光泽. 一种基于 Ethernet 的硬实时通信协议 [J]. 计算机研究与发展，2002，37(10)：19－23.

[6] Krishna C M, Kang G S. Real-Time System [M]. 北京：清华大学出版社，2001. 45(25)：24－29.

[7] Baker T. P. A Stack-Based Resource Allocation Policy for Realtime Peocessors [A]. Proc IEEE Real-Time System Symp [C]. Oakland: IEEE Computer Press, 1990: 191－200.

Study and Analysis of the Field Hard-Real-Time Networks Scheduling Algorithm

HU　Yong[1]，　YE　Ming[2]

1. College of Application Techaology, Chongqing Jiaotong University, Chongqing 400042;

2. School of Compnter and Information Engineering, SouthWest University of China, Chongqing 400715

Abstract: To meet scheduling condition of real-time tasks its characteristics of time are restricted in real-time systems. The real-time scheduling conditions are simplied when the people have studied this question. Scheduling condition and algorithm analysis of task sets are introduced, which task deadline was not less than its period. Slackening this restrict conditions, it adapted to scheduling of mixed model of period and no-period task. At the same time, scheduling restrict conditions of original RM algorithm is analyzed and shortcoming of computing time complexity is pointed out. Thus a new real-time scheduling approach and implementing flow-chart based on real-time networks scheduling algorithm is presented and given a strict contrast test of being put up real-time networks scheduling approach.

Key words: real-time system; networks scheduling; monotonic deadlines; scheduling times; scheduling speedup

责任编辑　张　枸