基于时间需求迭代和排队模型的开放式实时系统可调度性分析算法研究

牛 云 戴冠中 梁亚琳
(西北工业大学自动化学院 西安 710072)

摘 要 基于RM调度策略和可延期服务器调度的开放式实时系统,以往的可调度性分析算法造成较低资源利用率。结合时间需求分析和服务台休假M/M/1/K排队模型,考虑带宽保留服务器,提出一种高资源利用率的可调度性分析算法,对系统中所有周期任务进行可调度性分析测试,给出其在临界点的响应时间;根据非周期事件到来率和接收缓冲定量分析非周期事件的平均响应时间和事件丢失率。实验表明,提出的可调度性分析方法通过估计任务的响应时间范围,能够在较高资源利用率下,验证多任务系统的可调度性。

关键词 开放式实时系统,可延期服务器,时间需求分析法,服务台休假的M/M/1/K排队模型,可调度性
中图法分类号 TP316.2

Real-time System Temporal Parameters Analysis via Queueing Theory and Time-demand Analysis Method

NIU Yun DAI Guan-zhong LIANG Ya-lin
(College of Automation, Northwesten Polytechnical University, Xi'an 710072, China)

Abstract A new schedulability analysis method to calculate the schedulability of hard real time tasks and the general response time of soft real time tasks in open real-time system based on Rate Monotonic (RM) scheduling, while deferrable server was proposed. Time-demand analysis method was used to handle with hard real time tasks and the server vacation M/M/1/K queuing model was used to analyze soft real time tasks. Finally the schedulability of the whole tasks was calculated to make sure all the tasks can satisfy their deadlines and the experiment show that the calculate results of the analysis model accord with the system real status.

Keywords Open real-time systems, Deferrable server, Time-demand analysis, Server vacation M/M/1/K queuing model, Schedulability

1 引言

随着嵌入式系统的不断深入与发展以及Internet与嵌入式系统的相互结合,多类型的硬实时、软实时与非实时任务共存同一系统的情况越来越广泛,这种系统称为开放式实时系统[1],例如应用于工业控制现场的嵌入式Web服务器系统(EWS)。系统中工业控制任务具有硬实时性,而嵌入式Web服务提供工业控制现场运行的状态数据,没有硬实时性要求,需要与现场控制任务并行执行,同时不能影响实时性能。这种混合系统硬实时、软实时与非实时任务在时间需求上有不同特征,对这类混合任务集进行合理调度是保证系统实时性、正确性的关键。Z. Deng等提出的开放式实时系统调度的目标[1]:保证硬实时任务满足其截至期的基础上减小软实时任务的平均响应时间,同时系统中非相关的实时或非实时应用可独立进行实时性的验证测试。

传统的实时调度算法有动态和静态之分,分别以EDF和RM为代表[3,7]。一般来说基于EDF可以获得较高的资源利用率,但任务执行时间的可预测性差,在系统存在过载时,算法会急剧下降,不适合强实时任务。RM算法在任何情况下,任务的时间行为都可预测,适合调度强实时任务,但不能直接调度非周期任务。为此,采用基于带宽保留服务器[1,4-6]的资源预留方法。允许硬实时、软实时和非实时任务共存于一个系统,提供不同性质任务之间的资源隔离支持。

可调度性分析是根据特定的调度算法,验证给定的实时系统是否可以满足时间需求。调度算法必须要有与之相适应的可调度性分析算法才能实际应用。传统的RM可调度验证算法造成的资源利用率过低,且不能给出非周期任务的平均响应时间。

本文以本实验室自行开发的嵌入式智能节点服务器系统OpenWeb为实验对象,使用RM结合可延期服务器的调度算法,提供强实时任务与弱实时任务间的时间特性隔离。综合利用时间需求分析方法和服务台休假的M/M/1/K排队模型分析系统中强实时任务的可调度性、弱实时任务的平均响应时间。

到稿日期:2008-01-30 本文受国防基础科研项目(项目编号:C2720061361)资助。
牛 云(1980—),男,博士研究生,主要研究方向为嵌入式系统、实时系统调度算法,E-mail:niuyun03609@hotmail.com, ny036090331@tom.com;**戴冠中**(1937—),男,教授,博士生导师,主要研究领域为自动控制、信息安全;**梁亚琳**(1982—),女,硕士研究生,主要研究方向为计算机测控技术。

2 OpenWeb 系统任务模型

2.1 OpenWeb 系统的构成

本系统作为网络化测控系统的智能节点可以对多个对象进行检测控制。针对工业设备的多样性和分布性，本系统带有工业现场总线接口。通过现场总线访问下位机采集被控设备的状态，运行某种控制算法计算输出量，通过现场总线输出到执行机构。系统具有 Internet 接口，远程工作站可以通过 Web 请求远程检测设备的性能以及运行情况。

系统软件平台移植 VxWorks5.5 实时操作系统[10]，配置 GoAhead 嵌入式 Web 服务器[11]。利用 VxWorks 的任务管理 API，Hook 函数以及软件看门狗，定时器等开发中间件实现 RM 调度算法以及可延期服务器(DS)，并接管 Goahead 建立的 Web 请求处理任务，由 DS 调度管理。使调度算法对上层应用程序透明。

2.2 OpenWeb 系统任务划分

根据 OpenWeb 的功能，将应用软件划分为 6 个任务：

1. 现场总线通信任务(tCANCom)：周期访问现场总线上的节点设备，获得现场的运行状态采样，存入实时数据库。

2. 控制输出计算任务(tConOutput)：根据实时数据库中状态数据，判断系统的运行情况，利用特定控制算法计算控制输出值，送给执行机构。

3. 实时数据库管理任务(tRTdatabase)：更新实时数据时间戳，剥离过期数据，保证实时数据库中数据的时间/逻辑一致性。

4. Web 请求处理任务(tWebHan)：该任务监听 Web 请求，并根据不同的请求类型建立子任务进行请求应答。这些子任务由 DS 管理调度。

5. 运行状态历史数据记录任务(tHistroy)：周期记录现场设备的运行状况，方便系统维护。

6. 机内自检测任务(tBIT)：周期进行系统自检，保证智能节点的正常运行。

显然，该智能节点是开放式系统，硬实时、软实时任务并行运行，其计算结果的有效性与时限错过时间的关系如图 1 所示。实时输出控制指令的任务具有强实时性，一旦错过时限，会造成系统控制性能下降，甚至破坏系统稳定(有效性函数变为负值)。实时数据库管理任务如果错过时限有可能破坏实时数据的时间一致性，使控制算法依据的数据过时，属于强实时任务。Web 请求服务任务，错过时限不会造成灾难性后果，但过长的时间延迟会使远程监控得不到及时的系统运行信息，并可能使远程操作员作出错误判断。

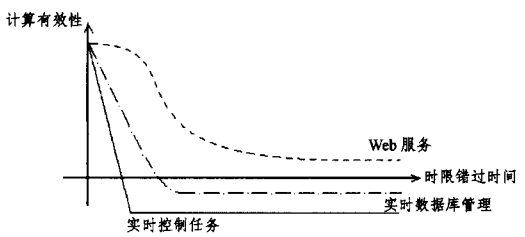

图 1 不同类型任务的计算结果有效性

2.3 OpenWeb 智能节点系统实时任务模型建立

为便于定量分析该混合任务系统的可调度性等实时性能指标，先对任务的时间参数进行建模。

设任务集

$$s=Tp_1,Tp_2,\cdots,Tp_n;Tap_1,Tap_2,\cdots,Tap_m \quad (1)$$

其中，$Tp_i(1\leqslant i\leqslant n)$为周期任务，每个周期任务 Tp_i 可用式(2)所示的四元式表示：

$$Tp_i=\langle Cp_i,Pp_i,\Phi p_i,Dp_i\rangle \quad (2)$$

其中，Cp_i 为任务的最大计算时间，Pp_i 为任务周期，Φp_i 为任务释放时刻，Dp_i 为任务时限；

$Tap_j(1\leqslant j\leqslant m)$为非周期任务，每个非周期任务可由如下的三元式表示：

$$Tap_j=\langle Cap_j,\Phi ap_j,Dap_j\rangle \quad (3)$$

其中，Cap_j 为任务最大计算时间，Φap_j 为任务释放时刻，Dap_j 为任务时限。

可延期服务器(Deferrable Server：DS)[4]，是带宽保留算法的一种，用一个或几个专用的具有较高优先级的周期任务管理所有非周期任务，使不同性质任务之间的资源隔离，减小相互影响。

延期服务器可用如下模型来描述：

$$T_s=[e_s,P_s] \quad (4)$$

其中，e_s 为服务器各周期的执行预算，p_s 为服务器执行预算的补充周期。$\rho_s=e_s/p_s$ 称为可延期服务器的强度。

上述任务调度模型如图 2 所示。

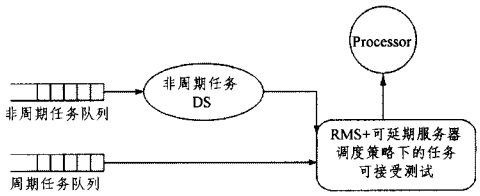

图 2 系统调度模型

根据以上模型，OpenWeb 任务集的时间参数如表 1 所示。其中，周期任务的时限等于其周期，执行时间在任务单独执行时测出，释放时刻为从 0 时刻起的周期的整数倍，表中不再表述。

表 1 OpenWeb 任务集时间参数表

任务名	属性	周期 (Pp_i/P_s)	执行时间 (Ct_i/e_s)	相对时限 (Dt_i)	优先级
DS	周 期	20ms	2.3ms	20ms	110
tCANCom	周 期	30ms	6ms	30ms	120
tRTdatabase	周 期	40ms	3ms	40ms	130
tConOutput	周 期	50ms	8ms	50ms	140
tHistroy	周 期	90ms	10ms	90ms	150
tBIT	周 期	120ms	26ms	120ms	160

3 周期任务可调度性测试

3.1 周期任务的临界时刻

RM 调度下，如果任务的某个作业与所有高优先级的任务同时释放，就发生该任务的临界时刻。该作业具有任务所有作业响应时间的最大值。

3.2 OpenWeb 混合任务集可调度性验证分析

对于具有硬截止期的周期任务，在 RM 下，多数采用式(5)式来检验有 n 个任务集的可调度性。但是式(5)是 RM 下任务可调度的充分条件，不满足式(5)任务集也有可能是可调度的。利用式(5)造成资源利用率偏低。所以，我们采用时间需求分析法分析其可调度性。

$$\sum_{i=1}^{n}U_i\leqslant n(2^{1/n}-1) \quad (5)$$

时间需求分析法由 Lehozky 首先提出，后来由 Audsley 等人补充完善[12,13]。但文献[12,13]未考虑开放系统下有带宽服务器时的情况，本文对此进行了改进。算法基本思想是估算各周期任务实例在临界时刻的执行时间需求并与任务时限相比较，如果系统中所有任务实例的时间需求都小于各自的时限，则系统可调度。该方法可以在不确定任务释放参数 Φp_i 的情况下估算出任务在临界时刻的完成时间。

若系统中可延期服务器具有最高优先级，上述验证方法的执行过程如下：

1. 按照(6)式计算 Tp_i 的时间需求函数

$$w_i(t)=Cp_i+e_s+\left\lceil\frac{t-e_s}{P_s}\right\rceil\times e_s+\sum_{T_k\in H_i}\left\lceil\frac{t}{Pp_k}\right\rceil\times Cp_k \qquad (6)$$

其中，H_i 为任务 Tp_i 的实际执行时间 t_i 内优先级高于 Tp_i 并且已就绪的任务集，T_k 为 H_i 中的任务，Pp_k 和 Cp_k 是 T_k 的周期和执行时间。

2. 检查不等式

$$w_i(t)\leqslant t \quad (t<=Dp_i) \qquad (7)$$

是否成立。t 是以被测任务的临界时刻为起点的时间区间。因为较高优先级任务的就绪或服务器的补充使 $w_i(t)$ 上升，此时被测任务的时间需求最大，若此时系统能够提供多于任务需求的时间，即存在一些区间使(7)式成立，则任务可调度。所以这里只需检验有高优先级任务就绪的时间点，即：$t_{x(j,k)}=jPp_k;k=1,2,\cdots,i;\ j=1,2,\cdots,\lfloor \min(Pp_i,Dp_i)/Pp_k\rfloor$，以及 $e_s,e_s+p_s,e_s+2p_s,\cdots,e_s+\lfloor(Dp_i-e_s)/p_s\rfloor p_s$。$Pp_k$ 为优先级高于 Tp_k 的任务执行周期。

更进一步，若有 $t_{x-1}<w_i(t)<t_x$，则有 Tp_i 的最大响应时间小于 $w_i(t)$。

用 q 表示系统中最大周期任务与最小周期的任务的周期比率，则上述验证算法的计算复杂度为 $O(nq)$，n 为任务数。

表 2 给出了 OpenWeb 任务集分析结果(仅列了 $t_{x-1}<w_i(t)<t_x$ 时刻的计算结果)。当可延期服务器执行预算大于 2.3ms 时，机内自检任务可能错过其时限。则我们可以得到在使用 RM 调度时，OpenWeb 系统在保证强实时任务满足时限的条件下，非周期任务可延期服务器的强度最高为 $\rho_s=e_s/P_s=2.3/20=11.5\%$。此时的 CPU 利用率为：87.78%。已不满足(5)式的要求，但实验证明该任务集是可调度的。

表 2 OpenWeb 任务集可调度性分析结果

任务名	时间需求分析 测试区间 t_x(ms)	时间需求 $w_i(t)$ (ms)	任务响应时间 (ms)	任务时限 (ms)	可调度性结论
DS	2.3	2.3	2.3	20	可调度
tCANCom	2.3	8.3	<=10.6	30	可调度
	20	10.6			
tRTdatabase	2.3	11.3	<=13.6	40	可调度
	20	13.6			
tConOutput	22.3	23.9	<=23.9	50	可调度
	30	23.9			
tHistroy	42.3	42.9	<=39.9	90	可调度
	50	45.2			
tBIT	102.3	116.8	<=119.1	120	可调度
	120	119.1			

4 非周期任务平均响应时间、请求丢失率分析

4.1 OpenWeb 系统非周期任务处理排队模型

本系统的非周期任务主要是 Web 请求的处理。利用 VxWorks 的 netbufLib 机制，为 OpenWeb 系统在协议栈应用层开辟队长为 K 的缓冲队列处理 Web 访问请求。当队列非空，且接管 Web 请求处理的 DS 有预算时，该请求被响应；队列满时，说明此时 OpenWeb 系统可能没有足够的能力在规定的时间内响应请求，该请求被丢弃。请求的产生符合泊松过程，通过实验求得分布参数的最大似然估计为 λ；响应服务独占 CPU 时，其执行时间 α 服从指数分布，本系统中服务受 DS 控制，需要根据 DS 预算的补充消耗规则计算广义执行时间；此外系统队长为 K 的有限长队列，服务方式服从先来先服务。因此，采用服务台有休假的 M/M/1/K 排队模型[14]分析系统中非周期任务的平均等待时间，请求丢失率。

4.2 广义执行时间的计算

本文中，DS 处理 Web 请求的广义执行时间指请求开始接受服务到服务完成的时间，其中包括 DS 预算耗尽的等待时间，记为 $\tilde{\alpha}$。文献[5]指出，如图 3 所示，在最坏情况下，DS 等价于执行时间有 e_s 抖动的周期任务。因此，不能简单地认为 DS 得到了 ρ_s 的 CPU 带宽。我们认为，考虑延期的执行预算，DS 实际的 CPU 带宽应由方程(8)确定。显然，当 $(n-1)e_s\leqslant\alpha\leqslant ne_s$ 时，UTL_{DS} 基本保持不变；$\alpha\to\infty$ 时，$UTL_{DS}=\rho_s$。

$$UTL_{DS}=\alpha/(\lceil(\alpha-e_s)/e_s\rceil\times p_s+e_s) \qquad (8)$$

本文基于服务执行时间 α 的概率分布，通过全概率公式计算概率加权的 DS CPU 带宽 $\widetilde{UTL}_{DS}$。将 α 按 e_s 的倍数划分为如表 3 所示的时间区域。

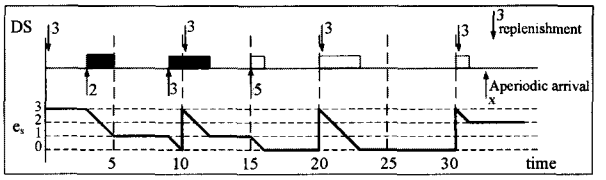

图 3 DS 的执行时间抖动

表 3 α 执行时间区间

α_i(ms)	<2.3	2.3~4.6	4.6~6.9	……	>92
UTL_{SDi}	1	0.187	0.147	……	0.115
$P_{\alpha i}$	0.108	0.097	0.086	……	0.011

由全概率公式(9)，得到 DS 实际的 CPU 带宽为 23.41%，比 ρ_s 高，这是 DS 延期的 e_s 造成的利用率抖动。

$$\widetilde{UTL}_{DS}=\sum_i UTL_{DSi}\times P_{\alpha i}=23.41\% \qquad (9)$$

于是：

$$\tilde{\alpha}=\widetilde{UTL}_{DS}\times\int_0^{+\infty}tdG(t) \qquad (10)$$

其中，$G(t)$ 为服务时间的分布函数。令，$\mu=\frac{1}{\tilde{\alpha}}$。

4.3 平均响应时间，丢失率的分析计算

根据带休假的 M/M/1/K 混合排队系统，$N(t)$ 表示广义执行时间下的 $0-t$ 时间段内系统 Web 请求队列的长度，系统的瞬时状态转移如图 4 所示，$\{N(t)\}$ 是一个嵌入马尔可夫过程。

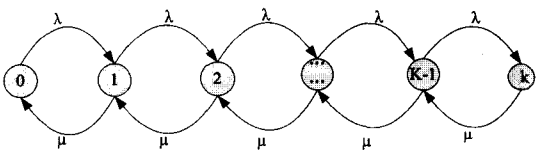

图 4 Web 请求处理排队模型状态转换

令 $\rho=\frac{\lambda}{\mu}$，称为排队系统的服务强度，有：

$$p_n=\rho^n p_0 \quad n=0,1,2,\cdots,k \tag{11}$$

$$p_0+p_1+\cdots+p_n=1 \tag{12}$$

(11)和(12)联立，可求得 $N(t)$ 的分布率：

$$p_n=\frac{(1-\rho)\rho^n}{1-\rho^{k+1}} \quad \rho\neq 1 \quad n=0,1,2,\cdots,k \tag{13}$$

对式(13)求期望获得平均等待队长为：

$$\bar{N}=\frac{\rho}{1-\rho}-\frac{(k+1)\rho^{k+1}}{1-\rho^{k+1}} \quad \rho\neq 1 \tag{14}$$

式(13)中，当 $n=k$ 时，得到的 p_k，表示系统对异步事件的损失概率，那么，单位时间平均进入系统的异步事件数 $\bar{\lambda}$ 为：

$$\bar{\lambda}=\lambda(1-p_k)=\frac{\lambda(1-\rho^k)}{1-\rho^{k+1}} \quad \rho\neq 1 \tag{15}$$

由式(14)和式(15)，再根据 Little 公式[14]，求得缓冲区中的异步事件消息从到达到处理完毕离开的平均响应时间 e_{ns} 为：

$$e_{ns}=\frac{\bar{N}}{\bar{\lambda}}=\frac{1}{\mu-\lambda}-\frac{k\rho^k}{u(1-\rho^k)} \quad \rho\neq 1 \tag{16}$$

5 OpenWeb 任务集的实际调度测试

图 5 是利用 VxWorks 提供的逻辑分析仪 WindView 观测绘制的系统任务集调度表。图 7－11 为 OpenWeb 系统各任务的响应时间分布图。以图 9 为例，该任务的响应时间主要分布在 8,11,13,14,17,20,23ms，由图 5 可知，上述响应时间分别对应任务不被抢占时，被 DS，tCANCom，tRTdatabase 分别或同时抢占时的情况。任务的响应时间不受非周期任务负载(Web 请求，图 6 所示)的影响，系统实现了强实时任务与弱实时任务在时间特性上的隔离。任务的最大响应时间符合 3.2 节验证算法的计算结果。实验说明，按照本文提出的可调度性验证算法进行任务准入控制，OpenWeb 任务集是可以调度的，且可以在负载不确定的开放性实时系统环境下，较

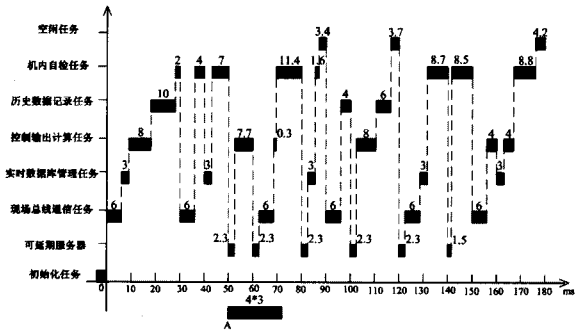

图 5 OpenWeb 任务调度表

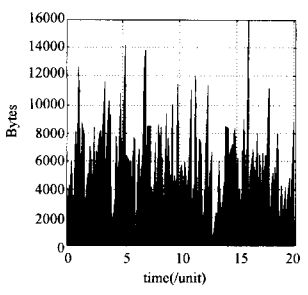

图 6 Web 请求流量

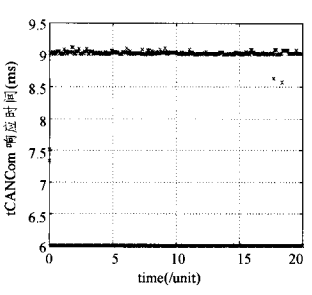

图 7 tCANCom 响应时间分布图

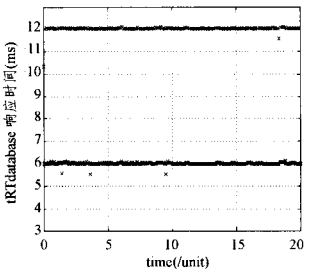

图 8 tRTdatabase 响应时间分布图

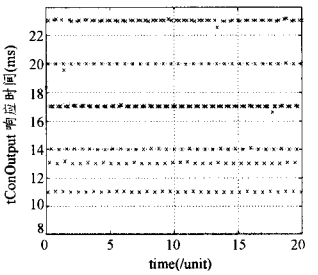

图 9 tConOutput 响应时间分布图

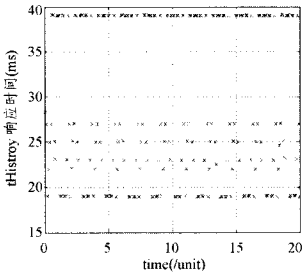

图 10 tHistroy 响应时间分布图

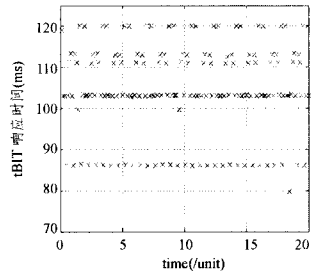

图 11 tBIT 响应时间分布图

为准确地估计出任务的最大响应时间范围。表 4 和表 5 分别显示了利用式(16)和式(13)估计的 OpenWeb 系统在一定缓冲下非周期任务的平均响应时间以及请求的丢失率。随着请求到来率的增加即样本容量的增加，算法估计值与运行值之间的误差逐步减小。

表 4 Web 请求平均响应时间

到来率	理论计算值	实际运行值	相对误差
λ=5	0.177	0.2071	17.01%
λ=10	0.4224	0.4564	8.04%
λ=15	0.8538	0.8267	3.17%
λ=20	0.9916	0.9804	1.13%

表 5 Web 请求丢弃率

到来率	理论计算值	实际运行值	相对误差
λ=5	0.00	0.00	0.0%
λ=10	0.0212	0.0255	20.28%
λ=15	0.2270	0.2406	5.99%
λ=20	0.4152	0.4232	1.92%

结束语 本文以实时任务时间需求分析法和服务台休假的 M/M/1/K 排队模型为分析工具，以嵌入式智能节点服务器系统 OpenWeb 为分析实例，建立了该系统的实时任务调度模型，得到了强实时任务最大执行时间范围和弱实时任务平均响应时间的数学表达形式，进而定量分析了系统的实时性能指标，给出了开放式实时系统可调度性测试算法。实验数据表明本文提出的测试算法的计算结果与系统实际运行结果相符。

参 考 文 献

[1] Deng Z, Liu JWS. Scheduling real-time application in open environment[J]//Proceedings of the 18th IEEE Real-Time Systems Symposium. Los Alamitos, CA: IEEE Computer Society Press, 1997:308-319

[2] McCombie B. Embedded Web servers now and in the future[J]. Real-Time Magazine, March 1998(1):82-83

[3] Wittenmark B, Nilsson J, Torngren M. Timing problems in real-time control systems // Proc. The 1995 American Control Conference. Seattle, Washington

[4] Liu J W S. Real-Time Systems[M]. Published by Higher Education Press arrangement with the original publisher, Pearson Education, Inc. , Beijing, 2002: 195-218

[5] Bernat G, ABurns. New results on fixed priority aperiodic server [C] // Proc. of the 12th IEEE Real-Time Systems Symposium. Phoenix, Arizona: IEEE Computer Society Press, 1999: 68-78

[6] Lin Suzhen , Manimaran G . A FeedBack - Based Adaptive Algorithm for Combined Scheduling with Fault-Tolerance in Real-Time Systems [J] // Proc. Conference on High Performance Computing (HiPC). Bangalore, India, Dec. 2004: 101-110

[7] Lu C, Stankvoic J A. Design and Evaluation of a Feedback Control EDF Scheduling Algorithm[J] // IEEE Real-Time Systems Symposium. Phoenix, AZ, Dec. 1999

[8] Abeni L , Buttazzo G . Intergrating multimedia applications in hard real-time systems[J] // Proc. 19th IEEE Real-Time SystemsSymposium. Madrid, Spain

[9] Cervin A , Eker J . Control - Scheduling Codesign of Real - Time Systems: The Control Server Approach[J] // Proc. Journal of embedded computing. 2004

[10] VxWorks-Programmer's Guide 5. 5, 2002 Wind River Systems [EB/OL]. NC. http://www. windriver. com

[11] Goahead Software Foundation[EB]. http://www. goahead. com

[12] Lehoczky J P. Fixed priority scheduling of periodic task sets with arbitrary deadlines // Proc. of the IEEE Real-time Systems Symposium. Dec. 1990

[13] Audsley N, Burns A, Tindell K. Applying a new scheduling theory to static priority preemptive scheduling. Software Engineering Journal, 1993, 5(5): 284-292

[14] 唐应辉，唐小我. 排队论基础与分析技术[M]. 北京：科学出版社，2006：57-61

（上接第 110 页）

用户交互操作的不可预知性和过于频繁的 VCR 操作，会使系统组播树处于剧烈振荡的状态。下一步研究将着重于使节点的 VCR 操作引起的组播树结构变动限制在一个较小的范围内，以改善组播树性能。另外，无线终端的普及也有力地推进了无线流媒体研究，可以将本文的研究工作推广到无线网络。

参 考 文 献

[1] Deering S. Host extendion for i Pmulticasting. RFC－1112. August 1989

[2] Quinn B, Almeroth K. IP multicast applications: Challenges and solutions. Internet Engineering task Force (IETF) Internet Draft, March 2001

[3] Chu Yang-Hua, Rao S G, Zhang Hui. A case for end system multicast // ACM SIGMETRICS. 2000: 1-12

[4] Jannotti J, Gifford D K, Johnson K L. Overcast: Reliable multicasting with an overlay network // USENIX Symposium on Operating System Design and Implementation. San Diego, CA, October 2000

[5] Guo Y, Suh K, Kurose, et al. P2Cast: P2P patching scheme for vod service// WWW2003. May 20-24

[6] The ns manual. July 23 2003. http://www. isi. edu/nsname/ns/

[7] GT-ITM. http://www. cc. gatech. edu/fac/Ellen. Zegura/graph. html

[8] Deshpande H, Bawa M, Garcia-Molina H. Streaming live media over a peer-to-peer network// Submitted for publication. 2002

[9] Padmanabhan V N, Wang H J, Chou P A, et al. Distributing streaming media content using cooperative networking // ACM/IEEE NOSSSAV. Miami, FL, USA, May 2002

[10] Sheu S, Hua K A, Tavanapong W. Chaining: A generalized batching technique for video-on demand // Proc. of the IEEE Ith'l Conf. on Multimedia Computing and System. Ottawa, Ontario, Canada, 1997: 110-117

[11] Castro M, Druschel P, Kermarrec A M, et al. SplitStream: High-bandwidth content distribution in cooperatie environment // IPTPS'03. Berkeley, CA, USA, 2003

[12] Tran D, Hua K, Do T. ZIGZAG: an efficient peer-to-peer scheme for media streaming // Proc. of IEEE INFOCOM'03. San Francisco, CA, USA, April 2003

[13] Diot C, Dabbous W, Crowcroft J. Multipoint communication: A Survey of protocols, functions, and mechanisms. IEEE Journal on Selected Areas in Communications, 1997, 15(3): 227-190

[14] Wei L, Estrin D. A comparison of multicast trees and algorithms // IEEE INFOCOM. Toronto, Canada, June 1994

[15] Waitzman D, Partridge C, Deering S. Distance Vector Multicast Routing Protocol. RFC 1075. 1998

[16] Moy J. Multicast Extension to OSPF. RFC 1584(1994)

[17] Ballardie J, Francis F, Crowcroft J. Core Based Trees // Proceedings of the ACM SIGCOMM. San Francisco, 1993

[18] Estrin D , Farinacci D , Helmy A , et al . Protocol Independent Multicast-Spare Mode (PIM-SM): Protocol Specification. Proposed Experimental RFC. Sept, 1996

[19] Padmanabhan V N, Wang H J, Chou P A, et al. Distributing Streaming Media Content Using Cooperative Networking // NOSSDAV 02. Miami, Florida, USA, May 2002

[20] Fracis P. Yoid: Extending the Internet Multicast Architecture. White paper. http://www. aciri. org/yoid, April 2000

[21] annotti J, Gifford D K, ohnson K L, et al. Overcast: Reliable Multicasting with an overlay network // USENIX Symposium on Operating System Design and Implementation. San Diego, CA, Oct. 2000

[22] Deshpande H, Bawa M, Garcia-Molina H. Streaming Live Media over a Peer-to-Peer Network. Technical report. Stanford University, 2001

[23] Banerjee S, Bhattacharjee B, Kommareddy C. Scalable application layer multicast // Proc. of ACM SIGCOMM. August 2002

[24] Kwon J B, Yeom H Y. Providing VCR Functionality in Staggered Video Broadcasting. Transaction on Cosumer Electronics (SCI), 2002, 48(1): 41-48

[25] Almeroth K C, Ammar M H. On the Use of Mlticast Delivery to Provide a Scalable and Interactive Video-on-Demand Service. IEEE Journal of Selected Areas in Coummunication (NGC'99), Nov. 1999: 152-169