第 12A 期
2002 年 12 月

电 子 学 报
ACTA ELECTRONICA SINICA

Vol.30 No.12A
Dec. 2002

实时中间件的研究与实现

郭长国，王怀民，邹 鹏，苑洪亮
（国防科技大学计算机学院，湖南长沙 410073）

摘 要： 解决分布式应用日益增多的实时问题需要实时中间件.实时中间件是分布式实时应用的支撑平台，它为应用提供表达实时约束的手段，并提供调度请求的机制和多种控制资源的方法，从而为分布式应用实现执行时间的可预测性提供方法.本文给出了实时中间件的概念体系，在分析中间件的体系结构模型的基础上，提出了一个多级线程池实时中间件体系结构，该模型不仅具有较强的并发处理能力，而且可以控制网络和请求使用的各种资源，能够提高系统的时间可预测性.采用该体系结构，我们实现了一个遵循实时 CORBA 规范的实时 ORB，并通过详细的对比测试验证了多级线程池实时中间件模型的有效性.

关键词： 实时中间件；可预测性；实时 CORBA
中图分类号： TP311.52 **文献标识码：** A **文章编号：** 0372-2112 (2002) 12A-2094-05

Research and Implementation of Real-Time Middleware

GUO Chang-guo, WANG Huai-min, ZOU Peng, YUAN Hong-liang
(*School of Computer Science, National University of Defense Technology, Changsha, Hunan* 410073, *China*)

Abstract: Real-time middleware holds an important position in increasing distributed application development. As the basic platform, the middleware should satisfy suchlike requirements: expression methodology of real-time constraints, real-time request scheduling mechanism, means of resource controlling and enabling the predictability of program lifecycle. This paper makes three contributions to the design of real-time middleware system. First, the paper defines the architecture of real-time middleware system. Second, under the analysis of distributed middleware applications, the paper presents a multi-level thread-pool real-time middleware model. Compared with traditional ones, the model has three advantages: better concurrent potential, more capability on network and resource controlling and higher system behavior predictability. Finally, the paper presents a real-time ORB under the specification of OMG real-time CORBA. Detailed experiments and comparison shows the efficiency of multi-level thread-poll real-time middle-ware model.

Key words: real-time middleware; predictability; real-time CORBA

1 引言

建立在网络环境上的分布式实时系统涉及的环节更多，因此较之传统的实时系统更为复杂.比如在一个军事指挥系统中，雷达捕获目标信号，将目标信号传送到目标识别系统，目标系统识别目标，然后将识别的结果传送到决策系统，由决策系统指挥相应的作战单位采取行动[1].这是一个典型的分布式实时系统流程，每一个环节都要求很强的实时特性，是单一实时系统所不能独立完成的.当今主流的分布式应用系统都是建立在中间件的基础上，人们十分关心中间件技术能否支持分布式实时系统的开发.

实时中间件的研究是目前中间件领域的一个研究重点，由于分布式实时应用的复杂性，实时中间件也是一个研究的难点.目前主要的分布式中间件平台，如 OMG 的 CORBA[2] 和 SUN 的 J2EE 都制定了相应的实时规范.OMG 组织目前已经制定了实时 CORBA1.0 规范.SUN 目前也制定了实时 Java 规范[3].MITRE, NRaD, Tri-Pacific 和 Rhode Island 大学在美国海军资助下较早的展开了实时 CORBA 的研究[4]，但是他们的主要工作集中在分布式实时调度方面；TAO[5] 的主要研究工作则更多的关注实时 ORB(Object Request Broker).关于分布实时 Java，目前有很多相关的草案[6]待选择.概括而言，人们对实时中间件的概念、模型和机制等问题还缺少系统的分析和研究.

本文首先讨论实时中间件的概念和设计目标；在分析通用中间件体系结构的基础上，提出一个能够提供较强时间可预测性的多级线程池中间件体系结构模型；按照该模型，本文给出一个实时 ORB 的实验系统，并通过详细测试对比验证模型的合理性.

收稿日期：2002-06-06；修回日期：2002-09-12
基金项目：863 重点课题(No.2001AA113020)；国家自然科学基金(No.90104020)；国家重点基础研究发展规划项目(No.G1999032703)

2 实时中间件的概念及设计目标

何谓实时中间件,它与实时系统是怎样的关系? 这是研究实时中间件首先需要界定清楚的问题.

2.1 关于实时系统

从用户的角度讲,实时系统的实时性是指系统或者系统中任务的执行时间遵守某种时间约束的特性.

实时系统的时间约束典型的表述为:在 k 时刻(前)必须完成任务 a,或者:任务 a 必须在 n 个单位时间内完成,或者:在单位时间内,任务 a 必须(或至少)被执行 q 次,以及:任务 a 必须在任务 b 之前完成.具体采用那种表述方式,根据不同的应用需求而定.

按照主流的系统层次模型,一个典型的实时系统由支撑平台和应用任务两个部分组成,传统的实时系统的实时性取决于系统使用的硬件,操作系统,甚至编程语言等支撑平台的实时特性.比如传统的实时系统,除了要能够确定处理应用任务的时间外,还要确定所基于的实时操作系统各管理环节的处理时间,以及对系统资源的调度与控制能力.实时操作系统实时特性的强弱,会最终影响系统的实时特性.对于建立在中间件上的分布式实时系统来讲,系统的实时特性还受中间件实时特性的影响.概括而言,实时系统需要实时支撑平台的支持.

2.2 关于实时支撑平台

那么,何谓实时支撑平台?,如果一个支撑平台能够确定其系统管理任务的时间特性,并为实时应用任务的时间可预测性提供相应的使能机制和手段,则我们称该支撑平台为实时支撑平台.

一般来讲,实时支撑平台不直接关注某个实时系统的时间约束,而应该关注对系统任务的时间可预测性或可确定性的支持.

典型的实时支撑平台为实时应用提供如下机制:

①处理器调度机制:实时应用任务通过这种机制确定任务何时占有处理器,从而再根据任务本身的处理时间计算是否满足相关的时间约束.特别对多任务系统,处理器调度机制确定了任务占有处理器的顺序,从而使任务执行时间的预测成为可能.应用任务可干预的处理器调度机制是实时支撑平台的核心,典型机制是优先级抢先调度机制,其他的调度机制可以基于优先级实现.

②资源控制机制:应用任务在处理过程中,需要使用各种各样的系统资源,比如申请和释放内存,I/O,使用线程资源等等.为计算系统任务执行时间,实时支撑平台必须提供控制每种资源的时间消耗范围.

实时操作系统就是实时应用的支撑平台,它一般不直接关注应用的实时要求,比如:每秒采集 25 次数据.典型的实时操作系统提供具有抢先能力的固定优先级调度机制,提供在有限时间内响应中断,提供有限的线程上下文切换时间等等.实时应用需要合理利用支撑平台的这些机制,以达到每秒采集 25 次数据的实时要求.

2.3 关于实时中间件

实时中间件是分布式实时应用的支撑平台,它为分布实时应用的时间可预测性提供使能机制和手段.实时中间件除了具有通用中间件的功能以外,它应为应用提供表达实时约束和特性的手段,并提供多种细粒度的控制机制,使应用能够调节中间件的各个处理环节,比如网络,内存和处理器等等,从而为应用任务提供某种程度的时间可预测性.

基于请求/响应的交互模型的分布实时应用要预测请求执行的时间特性.一个请求涉及客户和服务方的处理和网络通讯,其中客户和服务方的处理涉及本地的一个或者多个任务.实时中间件的关键就是要提供相关实时机制来保证请求执行时间的预测性.在此背景下,实时中间件的设计需考虑以下三个方面的问题:

①控制处理器资源的使用:应用任务通过这种机制可以确定占有资源和请求被处理的时机,对预测请求的执行时间非常重要.同时,实时中间件必须有较强的并发和抢先能力,因为分布式服务必须同时服务于多个客户,而抢先能力对实时应用特别重要:实时应用必须对实时任务及时处理.

②控制网络资源的使用:网络是分布式应用中的重要资源,应用任务对网络的控制和使用对预测请求的网络通讯时间会产生重要影响.

③控制其他资源的使用:请求被处理过程中,还会涉及到其它资源的使用,比如线程资源,存储资源等等,对这些资源的控制,也将对预测时间产生重要影响.

实际上,一般中间件强调对用户透明,而实时中间件实质上是一种 QoS 机制,它必须在透明和可控制上取得平衡.

3 实时中间件模型

本节讨论支持请求/响应的交互模型的实时中间件.

3.1 简单中间件体系结构模型

简单的中间件体系结构是一种单线程结构,这种模型不适合并发实时服务的实施,因为这种 FIFO 的模型不区分请求的差异,所以难以预测请求被调度的顺序和占有处理器的时刻,因而难以预测请求的时间特性.

3.2 多线程中间件体系结构模型

分布式服务一般都要同时为多个客户提供服务,所以必须具有并发能力.典型的并发模型在请求的每个环节都派发一个线程,从而可以最大限度的利用处理器的资源.

在多线模型中,可以规定请求的优先级,处理该请求的线程使用请求对应的优先级,并且按照优先级高低调度线程,保证高优先级的请求先执行,从而提高系统执行顺序的可预测性.这种模型的缺陷是难以控制系统的存储器和网络资源,系统规模的可预测性较弱.比如如果请求到达得非常频繁,而请求的处理时间又比较长,这时,系统中的线程数量会急剧增加,甚至使系统无法正常工作.另外,在一个请求的执行过程中,涉及多个线程上下文切换,特别是线程都动态创建,而线程创建时间又依据系统负载差异很大,从而造成请求执行时间的可预测性较弱.同时,这种模型没有对网络提供细力度的控制.

3.3 实时中间件体系结构模型

为了提高中间件对分布实时应用的支持,需要使用新的

中间件体系结构模型，我们提出了图 1 的多级线程池模型，它可以从多个方面提高系统的时间可预测性.

实时中间件模型 M 可以表示如下：

$M=\langle C, T_r, T_m, Qr, B_e, Qp, T_s\rangle$，其中

C 表示网络连接集合，$C=\{C_1, C_2, \cdots\}$，$C_i(i=1,2,\cdots\cdots)$表示一条网络连接，网络连接的个数随服务的客户数变化，一个客户可以有一条或者多条连接.

T_r 表示接收线程池(集合)，$T_r=\{T_r^1, T_r^2, \cdots, T_r^{n0}\}$，$T_r^i(1\leqslant i\leqslant n_0)$表示一个接收线程.

T_s 表示发送线程池(集合)，$T_s=\{T_s^1, T_s^2, \cdots, T_s^{n1}\}$，$T_s^i(1\leqslant i\leqslant n_1)$表示一个发送线程.

Qr 表示请求接收队列. $Qr=\{F_1, F_2, \cdots\}$，F_i 表示请求. Qr 是 T_r^i 的共享资源，T_r^i 接收到请求后，在确定了优先级之后都加入 Q_r 中.

T_m 表示请求解码和适配(鉴别)线程池(集合)，$T_m=\{T_m^1, T_m^2, \cdots, T_m^{n2}\}$，$T_m^i(1\leqslant i\leqslant n_2)$表示一个请求鉴别线程.

$B_e=\{\langle Q_{e1}, T_{e1}\rangle, \langle Q_{e2}, T_{e2}\rangle, \cdots, \langle Q_{en3}, T_{en3}\rangle\}$ $Q_{ei}(1\leqslant i\leqslant n_3)$表示请求执行队列. 任一个 Q_{ei} 队列都和一组对象实现相关，队列中存放需要这些对象实现执行的请求，这些请求是请求鉴别线程对请求鉴别后放入的. $T_{ei}=\{T_{ei}^1, T_{ei}^2, \cdots, Tei^{n4}\}$，表示和一组对象实现相关的一组请求执行线程. 每一个请求执行队列 Q_{ei} 都对应一组请求执行线程 T_{ei}. 当 Q_{ei} 存在请求待执行时，它就按照请求的优先级依次从对应的 T_{ei} 获取线程执行请求.

Q^p 表示应答发送队列. T_{ei} 中的执行线程完成请求后，将应答送入 Q^p. 每当 T_{ei} 中有应答要发送时，它按照应答的优先级高低从 T_s 中选取线程，发送应答. 其工作流程如图 2 示.

首先，模型具有很强的并发能力和抢先能力，提供基于优先级的控制处理器的手段. 模型中，请求具有优先级，请求的每个处理环节都使用不同的线程，使模型有很强的并发能力. 请求的每个处理环节都使用优先级等待队列和线程池. 当请求所需的资源无法满足时，请求就在对应优先级的等待队列中等待，当有资源可用时，优先分配给高优先级等待队列中的请求. 这保证高优先级的请求可以在多个环节抢先低优先级的请求执行，这对预测请求的执行顺序，从而预测请求执行时间非常关键.

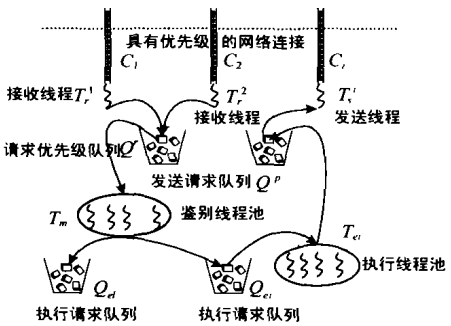

图 1 多级线程池中间件体系结构

其次，模型使用线程池控制系统规模. 因为在预测请求执行时间时，必须考虑请求执行过程中创建线程的时间和线程上下文切换的时间. 但是这两个时间受系统负载的影响很大，特别是负载急剧增加时，线程切换时间也急剧增大，最终变得不可预测. 使用线程池，一方面使得可以确定系统的最大规模，这样可以确定系统的负载，从而可以增强对线程上下文切换时间的可预测性；另一方面，线程池的线程在系统启动时即已经创建，所以预测执行时间时可以忽略线程的创建时间. 线程池中的线程的优先级，线程的堆栈都是可以控制的. 请求使用的线程数目达到最大时，请求就在对应的队列中等待，当有空闲的线程时，优先分配给高优先级的请求. 使用线程池可以很好地控制系统规模：系统中的最大线程个数是确定的，并且根据线程栈的大小和等待队列的控制参数可以估算系统中使用的静态存储器资源.

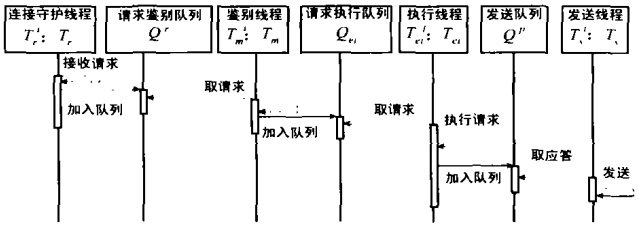

图 2 实时中间件的请求处理

本模型还提供很强的网络连接控制能力，在我们的模型中，使用三个手段控制网络连接，首先是控制网络连接的建立时间，其次是控制连接的复用属性，即可以为某些服务建立专有的连接，以保证请求能够及时发送. 第三是为网络连接规定优先级：不同优先级的请求，使用对应优先级的网络连接发送，保证请求的优先级贯串到请求的整个处理过程之中. 使用优先级的网络连接还可以针对具有优先级能力的网络协议进一步提高可预测性. 比如对 ATM 网络，就可以将请求的优先级映射到网络中，保证在网络传输过程中，高优先级的请求先传输，可以更好的提高可预测性. 对具有服务质量的 TCP/IP 网络，可以为高优先级的请求使用较高的 802.1p 优先级或者相应的 DSCP(DiffServ Codepoint)[7]，使它可以优先通过路由和传输设备，这些对提高可预测性都十分有益.

实时中间件的多级线程池体系结构通过为网络连接，请求接收发送以及处理队列和所有相关的线程规定优先级，在各个环节保证高优先级的请求优先被处理；通过细粒度控制多个队列和多级线程池的属性，保证系统的规模可控；通过网络和线程池，尽可能地减少了请求执行过程中的不确定性. 这些都增强了分布式实时应用的时间可预测性.

4 实时 CORBA 及其实现

我们遵循实时 CORBA 规范，使用多级线程池中间件体系结构，实现了实时 ORB. 和普通的 ORB 相比，实时 ORB 能够从多个方面提高系统的时间可预测性.

实时 CORBA 使用固定优先级调度方法，使请求者或者服务可以表达自己的优先级信息，实时 ORB 优先处理高优先级的请求，并且高优先级的请求可以剥夺低优先级请求. 一个请求的优先级可以是客户发送时规定的，也可以是服务事先声明的. 前者称为客户传播模式，后者称为服务声明模式. 为了能够控制请求执行的细节，实时 CORBA 从计算资源，网络资源和存储资源三个方面提供了控制方法. 在控制方法上，实时

CORBA 使用策略的方式:它规定了一组策略,并且提供设置这些策略的接口.

为控制网络,实时 CORBA 规定了网络相关的策略,比如连接建立时间,连接对应的请求优先级范围等等.在控制 ORB 资源上,实时 CORBA 规定了一组 POA 的策略,通过设置 POA 策略,可以控制线程池属性和系统中请求缓存的属性.

遵循实时 CORBA 规范,按照多级线程池体系结构,我们独立设计和实现了一个实时 ORB—RTS.RTS 具有多级的优先级队列,使得它具有很强的并发处理能力,能够对高优先级的请求作出及时和快速的反应.RTS 提供了多级的线程池,使其具有很强的资源控制能力.

在实时 CORBA 的实现中,我们将请求的处理过程尽量细分,每个处理环节尽可能短,并且每个环节都使用优先级队列,一方面增加系统的并发能力,一方面能够及时匹配请求的精确优先级.这保证 ORB 可以及时响应和处理高优先级的请求.另一方面,在每个环节都使用线程池,它的好处是可控制 ORB 的并发和资源使用规模,可根据应用特性调整 ORB 参数,比如对请求少但是执行时间长的应用就可以减少解码线程池的线程个数.这样达到提高系统可预测性的目的.

5　实时 CORBA 实现的对比测试

考虑一个分布式的数据收集系统.该系统中有 n 个分布的数据采集器 $C_1, C_2, \cdots, C_n (n \geqslant 1)$,它们将采集的数据上报给数据收集服务器 S.假设 S 具有每秒接受 $m(m \geqslant 1)$ 次数据上报的能力,每一个采集器希望每秒上报数据 $q(1 \leqslant q \leqslant m)$ 次.当 $nq \leqslant m$ 时,无论是实时 CORBA 还是非实时 CORBA,所有的采集器都能够每秒上报 q 次数据(假设它们每秒都能够发送 q 次请求);当 $n_q > m$ 时,对非实时 CORBA,没有一个采集器可以每秒上报 q 次数据.在 RTS 上,我们为每一个采集器分配了优先级,使得 S 能够满足高优先级的数据采集器的上报数据的要求.不失一般性,假设 $C_1, C_2, \cdots, C_n$ 的优先级 $P_1 \geqslant P_2 \geqslant \cdots \geqslant P_n$,将有 $k(1 \leqslant k \leqslant n, k_q \leqslant m, (k+1)q > m)$ 个优先级最高的采集器 $C_1, C_2, \cdots, C_k$ 可以每秒上报 q 次数据,优先级位于第 $k+1$ 个的采集器 C_{k+1} 将上报 $m - kq$ 次数据,其他的采集器 $C_{k+2}, \cdots, C_n$ 将几乎难以上报数据.这种能力正是实时中间件可预测性的体现.

在对比测试中,m 在 55 ~ 60 之间,$n = 5$, $q = 25$.试验记录了每个采集器每秒上报的数据和数据收集服务器每秒收集数据的次数,可以比较明显地看到非实时和实时服务的差异.

5.1　非实时 CORBA 的测试

在非实时 CORBA 中,服务为所有的客户提供相同质量的服务,在任意时刻,系统中的客户平均分享全部的服务能力.

C_1 在 1 时刻被启动,7 时刻启动 C_2,因为服务能力可以满足两个客户,所以 C_1 和 C_2 仍然获得全额服务;在时刻 14,C_3 被启动,这时服务难以胜任,所以 C_1、C_2 和 C_3 都只能得到非全额的服务,服务质量下降;在 29 时刻,C_4 被启动,服务质量再次下降;在 40 时刻,C_5 被启动后,服务质量降至最低.52 时刻 C_1 关闭后,$C_2C_3C_4C_5$ 服务质量得到提升,56 时刻 C_2 关闭后,$C_3C_4C_5$ 服务质量又一次得到提升,但是因为系统中还有三个客户,所以依旧难以达到全额服务.在 63 时刻当 C_3 关闭后,C_4 和 C_5 获得全额服务.

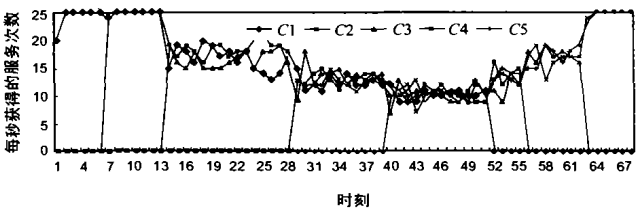

图 3　非实时 CORBA 所有采集器上报数据图

图 3 表明,所有客户共享全部服务,要么都获得全额服务,要么服务质量全部下降.

5.2　实时 CORBA 的测试

在时刻 1,启动优先级为 24 的 C_1,由于只有一个客户,所以 C_1 可以得到全额服务,即可以每秒上报 25 次数据;在时刻 4,启动优先级为 16 的 C_2,由于服务可以每秒服务 50 次以上,所以 C_2 和 C_1 都可以同时得到全额服务,即每秒都可以上报 25 次数据;在时刻 15,启动优先级为 30 的 C_3,这时服务无法同时满足 C_1、C_2 和 C_3,因为在它们中 C_2 的优先级最低,所以服务降低 C_2 的质量,保证优先级最高的 C_1 和 C_3 获得全额服务,C_2 只能获得剩余的服务能力,即每秒大约上报 8 次数据;在时刻 22,启动优先级为 27 的 C_4,同样的原因,服务只能为最高优先级的 C_3 和 C_4 提供全额服务,降低 C_1 的服务质量,C_2 几乎难以得到服务;在时刻 34,关闭优先级为 30 的 C_3,这时优先级最高的 C_1 和 C_4 将获得全额服务,所以 C_1 的服务被提升,每秒上报 25 次数据,相应的剩余服务能力被 C_2 获得;在时刻 46,关闭优先级为 24 的 C_1,这时只有 C_2 和 C_4 请求服务,所以它们都可以获得全额服务,即每秒上报 25 次数据;在 55 时刻,启动优先级为 22 的 C_5,这时服务将再次降低 C_2 的质量,保证优先级最高的 C_4 和 C_5 获得全额服务,C_2 只能每秒上报 8 次左右的数据;在时刻 66,关闭 C_4,这时系统中只有 C_2 和 C_5,所以 C_2 的服务再一次被提升,和 C_5 一起获得全额服务,并在时刻 75 时关闭.

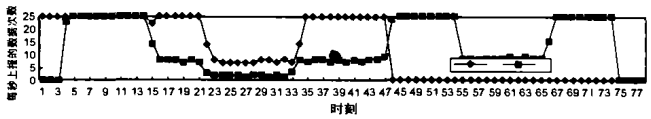

图 4　优先级 24 的 C_1,优先级 16 的 C_2 上报数据图

如图 5 所示,C_3 优先级在系统中始终最高,所以一直获得全额服务.C_4 优先级在系统中始终保持次高或者最高,所以总能获得全额服务.C_5 被启动以后优先级在系统中处于最高和次高,所以一直获得全额服务.

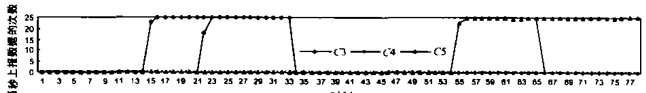

图 5　优先级 30 的 C_3,优先级 27 的 C_4 和优先级 22 的 C_5 上报数据图

如图 6 所示,当系统中的客户大于两个时,服务总是达到极限.并且可以看到在实时情况下的执行效率要高于非实时

情况.实时情况下满负荷时可以达到 58 ~ 59 次服务/每秒,而非实时情况下只能达到 51 ~ 53 次服务/每秒.

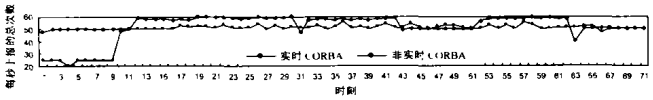

图 6 实时和非实时 CORBA 中的收集服务图

6 结论

本文从实时支撑平台的角度给出了实时中间件的概念,其核心是为分布式实时应用提供预测和控制执行时间的机制和手段,提出了实时中间件需要关注的三个问题,即处理器资源的控制、网络资源的控制和存储等其他资源的控制.本文讨论支持请求/响应的交互模型的中间件模型,提出了基于多级线程池的实时中间件体系结构,并采用该体系结构实现了一个具有固定优先级调度能力的实时 ORB.通过详细的对比测试和分析,我们认为本文对实时中间件的定位是合理的,所提出的实时中间件体系结构是有效的.

参考文献:

[1] B Kao, H Garcia-Molina. Deadline assignment in a distributed soft real-time system [J]. IEEE Transactions on Parallel and Distributed Systems, 1997, 8(12): 1268 - 1274.

[2] Object Management Group. The Common Object Request Broker Architecture and Specification, 2.4.1 edition[S]. 2000.

[3] The Real-Time for Java Expert Group. The Real-Time Specification for Java[S]. 2000.

[4] G Cooper, et al. Real-time CORBA development at MITRE, NRaD, tri-pacific and URI [A]. in Proceedings of the Workshop on Middleware for Real-Time Systems and Services [C]. San Francisco, CA, 1997. 69 - 74.

[5] D C Schmidt, D L Levine, S Mungee. The design of the TAO real-time object request broker [J]. Computer Communications, 1998, 21(4): 294 - 324.

[6] E Douglas Jensen. A proposed initial approach to distributed real-time Java[A]. in Proceedings of the Third IEEE International Symposium on Object-Oriented Real-Time Distributed Computing [C]. Newport Beach, California, 2000. 1 - 6.

[7] Yoram Bernet. Networking Quality of Service and Windows Operating Systems [M]. Indianapolis: New Riders Publishing, 2001.

作者简介:

郭长国 男,1973 年生于河南武陟县.分别于 1996 年 7 月和 1999 年 3 月在国防科技大学计算机学院计算机科学与工程专业获得学士和硕士学位,现为国防科技大学计算机学院博士研究生.研究兴趣是分布计算,数据库技术和智能软件.

王怀民 男,1962 年生于江苏南京市,教授,博士生导师,863 专家,主要研究方向为智能软件,分布计算和网络信息安全技术.

邹 鹏 男,1956 年生于山东高青县,教授,博士生导师,主要研究方向为操作系统和分布计算.