开放系统中实时中间件关键技术研究*)

彭 舰[1,2] 俞 岭[1] 董 鹏[1] 刘锦德[1]

(电子科技大学计算机学院微机所 成都610054)[1] (四川大学计算机学院 成都610065)[2]

Research on the Key Technology of Real-Time Middleware in Open System

PENG Jian[1,2] YU Ling[1] DONG Peng[1] LIU Jin-De[1]

(Computer Institute, Computer School of UEST, Chengdu 610054)[1] (Computer School, Sichuan University, Chengdu 610065)[2]

Abstract The Real-Time Middleware is the best way to deal with open and real-time in the same time in open system. In this paper, the characteristic of open system, the policy of real-time implement in open system and the real-time CORBA specification and development are introduced, then the key technology of real-time CORBA in open system is discussed in detail according to the model of real-time CORBA including the technology of the real-time POA, real-time ORB, thread pool, pluggable network protocol, multiplexing and demultiplexing and presentation layer optimizations etc, which provides a viable method and research for open system real-time extension.

Keywords Open system, Real-time, Middleware, CORBA, ORB, POA, Real-time scheduling, Thread pool, Network protocol

1. 引言

开放系统已经成为当前计算机界中的一个流行名词，对其相关领域的研究也成为热门。尽管对开放系统本身的定义并不统一和精确，但还是存在着公认的必具特征，即应具有：(1)可移植性；(2)可互操作性；(3)可伸缩性；(4)易获得性。

通过对这4个特征的细化分析和引申，可得出一个比较完善的开放系统的基本功能：(1)采用种种工业标准(或公认标准)；(2)具有应用可移植性；(3)具有最终用户可移植性；(4)具有软件开发者可移植性；(5)具有系统管理员可移植性；(6)具有可互操作性；(7)具有良好的安全性。

从上面开放系统的定义和功能可以看出开放系统更注重互操作性，而对实时这样的属性很少去考虑。

在实时系统中，对于任务的执行存在时间上的限制，为了满足时间上的限制，需要精确分析时间行为，调度和分配资源等，所以原有的实时系统不少采用专用系统来实现。随着实时性能的重要性日见突出，应用的实时性要求和系统的开放性已经成为了一对矛盾。本文将研究“开放”与“实时”之间的融合之道，着重阐述利用实时中间件构造实时开放系统，以及开放系统中实时中间件的关键技术。

2. 开放系统中实时的实现策略

实时性是开放系统技术发展的一个新方向，相关的研究较多，目前正从以下几个方向来研究和探索解决开放系统中的实时性的途径：

(1) 中间件方案 这一方案首先由长期从事开放系统技术研究的人员提出。在解决了开放系统的许多功能性难题(特别是互操作问题)之后，他们发现实时性将会成为开放系统的一个重要特征。在原有解决开放系统方案的基础上，进行了扩展以支持实时应用。最早的研究成果是APM的ANSAware/RT。利用它可以为用户建立一个高级别的开放系统的环境，在此环境中用户可以获得异种系统之间的可移植性和可互操作性。应该说，对现有中间件进行实时扩展是解决开放系统中的实时性的最终途径。

(2) 实时调度方案 这一方案主要是由从事实时调度研究的人员提出，他们希望通过新的调度机制来建立开放的实时环境。较有代表性的是：文[3]中提出的一个面向实时应用的开放环境结构，它在Windows NT内核中实现两级调度。在该系统中，每一个实时应用由一个服务器执行。在低一层上，操作系统的调度器按EDF算法调度所有的服务器。在高一层上，每个服务器的调度器调度应用作业，服务器按照应用选择的调度算法执行应用。这些研究希望通过对调度算法的改进，在操作系统级来建立“开放的实时环境”，离真正的开放系统还有很远的距离。

(3) 实时QoS方案 这一方案首先由DARPA提出，是在中间件中引入QoS机制，通过该机制的灵活性，向实时应用提供必要的资源，以确保实时任务在一个高度动态、难于预测的环境中成功完成，同时，保证异种资源的良好伸缩性。

以上3个方案从不同的方向来探讨开放系统中的实时性的某一个方面，但都不全面，还没有上升到开放系统的高度来看待实时性。但是这些研究也向我们指明了开放系统中的实时性的研究方向，那就是实时中间件是解决开放系统中的实时性的最终途径，但是必须辅之以实时调度和实时QoS的支持。

3. 实时CORBA规范及其发展

CORBA中间件作为中间件技术的一种，其技术研究领先性一直为业界所推崇。但在实时方向，OMG也是从1997年才开始发出有关实时CORBA的RFP。到1999年，终于决定正式采纳实时CORBA1.0的建议作为规范，随后，在

*)国防预研基金：新型系统体系结构技术(41315010103)；四川大学青年基金。彭 舰 博士生，讲师，主要研究方向：开放系统与中间件技术。俞 岭 硕士生。董 鹏 硕士生。刘锦德 教授，博士导师，主要研究方向：开放系统与中间件技术，网络多媒体技术，移动Agent技术等。

CORBA3.0规范中发布了与实时相关的规范。本节将重点简介实时 CORBA 中间件的规范及其发展。

固定优先级的实时 CORBA 模型如图1所示 。它由网络界面、实时操作系统、通信协议、实时 CORBA 中间件(实时 ORB 等)和实时调度服务所组成。为了实现系统的全局实时性,每一个部分都应当支持必要的实时性,对实时性有显著影响的方面主要包括:

·实时操作系统。ORB 实际上是利用操作系统的机制来调度应用级的活动。由于实时 CORBA 限于面向优先级固定的实时应用,因此实时应用的活动就对应于管理操作系统线程的优先级。所以实时 CORBA 着重考虑的是那些允许应用指定调度优先级和策略的操作系统。

·实时调度服务。实时操作系统提供的调度处于相对较低的层次。为了允许应用在较高层指定调度需求,实时 CORBA 规范定义了调度服务,该服务负责分配系统资源来满足实时应用的时间约束需求。通过调度服务的使用,应用中所涉及的低层实时构造的复杂性得以屏蔽。

·实时 ORB。ORB 负责透明地在客户和服务器之间传递请求。但对一个实时 ORB 应提供面向实时应用的标准界面,以便应用可以向 ORB 指定它们的资源需求,如线程优先级、消息排队的缓冲和传输层的连接等。实时 ORB 与实时调度配合,使应用执行时对其规定的端到端时间约束得以实现。

·网络和通信的资源管理。一个实时 CORBA 端系统必须具有能影响底层通信基础设施的策略和机制,以支持资源的保证。这一支持将涉及:(a)管理某个调用的连接的选择;(b)利用网络的高层次 QoS 特征,如控制结束连接的时间。

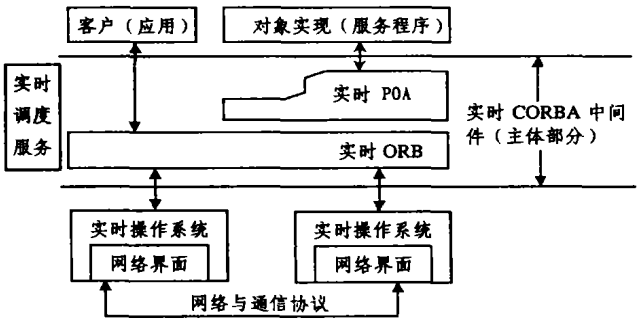

图1 实时 CORBA 的概念模型

为了管理这些方面,RT_CORBA 定义了标准的界面和 QoS 策略,以便应用配置和控制:

(1)处理器资源(通过线程、优先权机制、进程内互斥和全局调度服务);

(2)通信资源(通过协议属性和显式绑定);

(3)内存资源(通过缓冲要求和线程池大小)。

应用可通过调用标准的 ORB 操作,来指定这些实时 QoS 策略。实时 CORBA 定义了一个界面,允许应用指明 ORB 层和具体传输层的协议属性,以控制不同的通信协议特征,如 Internet 的 RSVP 的流速率。实时 CORBA 定义了一对 QoS 策略:ClientProtocol 和 ServerProtocol,用来选择和配置理想的 ORB 和传输协议属性。

本文的重点在于支持实时应用的中间件。为了引入实时性能,我们在研究和开发实时 CORBA 中间件时应对常规的 CORBA 做好一系列的实时扩展。如图2所示。这些扩展的具体内容为:

(1)将客户的 CORBA::Current 界面扩充为 RTCORBA::Current 界面;

(2)将服务器的 POA 扩充为 RTPOA(实时 POA);

(3)将 ORB 扩充为 RTORB(实时 ORB);

(4)增添 RTCORBA::Priority 界面和 RTCORBA::PriorityMapping 界面;

(5)增添 RTCORBA::Threadpool 界面;

(6)增添实时调度服务。

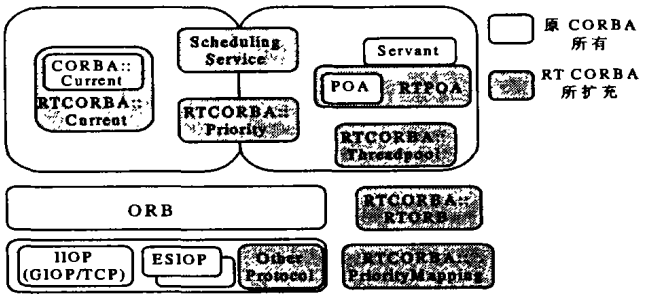

图2 实时 CORBA 扩展

4. 实时 CORBA 中间件关键技术的研究

从图1可以知道,要对中间件进行实时扩展,应该从所有可能的层面或方向着手。

1) 在应用层面上,要为实时 ORB 端系统设计适用于实时的应用界面

实时 ORB 端系统是实时 CORBA 系统中的主体部份,也是保证"实时应用中种种活动的行为具有端到端的可预测性"的物质基础。在实时 CORBA 系统中,ORB 负责透明地在客户(实时应用)与对象实现(服务程序)之间传递请求。这时每个实时 ORB 端系统应该提供合适的应用界面,以便客户(实时应用)在向 ORB 发出请求时,可以指明它对时间约束的需求,如死线、周期、执行时间、重要性等。这是实时 CORBA 系统务必具备的功能。

实时 CORBA 还提供了一个机制使客户能对服务器作显式绑定。同时该机制还允许用户指定要使用的网络协议。实时 CORBA 中客户使用了显式绑定,即用户在调用操作前根据实际的要求预先建立相应的连接。这样一方面减少了操作调用过程中的连接延迟和抖动,提高了操作调用的可预测性,另一方面能有效地分配和控制资源的使用。实时 CORBA 定义了两种策略来支持显式绑定:(a)优先级分段连接;(b)私有连接。

2) 在 ORB 内配置适用于实时应用的线程池

相对于一般 CORBA 中的 ORB,实时 CORBA 中 ORB 的特殊部分是线程池(如图3)。它是一个极为灵活的多线程服务提供者。当少量应用(或活动)同时到来时,它用已准备好的线程(静态线程)对等数地服务;当数量较多的应用(或活动)同时到来时,它就增派备用的线程(动态线程)来填补服务;当数量很多的应用(或活动)同时到来时,它就从它处(同池其它组)借用线程来增补服务;当数量过多的应用(或活动)同时到来时,它就让无法得到线程的应用在缓冲队列中等待。由于有了线程池可以承受大量应用(或活动)的并发,因此它有能力对付实时应用的需要。

正确设计 ORB 中的线程池十分重要,因为好的方案能:(1)缩短时间延迟;(2)减少延迟抖动;(3)防止优先级反转,有利于保证时间的确定性;此外还有利于节约资源消耗。

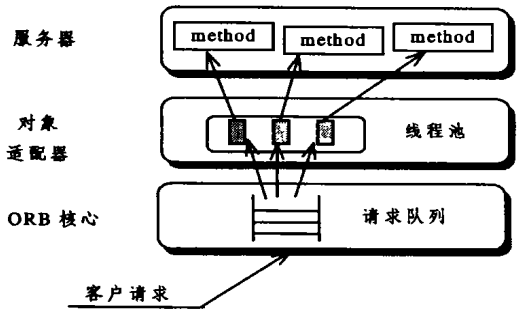

图3 线程池机制

3) 为 ORB 建立,对 C/S 连接通路可选择不同方案的措施

实时应用执行中最难防止的是优先级反转的发生,而且其后果引起的不确定性也较为严重。对于一般 CORBA 系统,因为较少关注时间因素而更多考虑的是效用,所以常采用多工技术,即通过复用(Multiplexing)和解复用(Demultiplexing),使一个连接通路为多个请求服务(以时分方式实现)。但在实时 CORBA 系统中若对 C/S 连接通路采用这一技术,会发生多种优先级的请求在同一连接通路上传送,最后,导致优先级反转经常发生。针对这一问题,应建立上层措施,以便用户(应用)可选用合适的 C/S 连接通路的具体方案。例如可选用为各优先级的请求派发专用的 C/S 连接通路的方案,这样就有可能防止上述优先级反转的发生。在此可以指出,在实时 CORBA 系统中合理的线程池方案和连接通路方案,是减少和防止优先级反转的两大重要措施。

4) 配置适用于实时应用的对象适配器

POA 的重点任务是配合 ORB 完成"请求处理"。在一般 CORBA 中 POA 在完成"请求处理"时将会遇到3次解复用的操作,这将带来很长的时间延迟和较多的优先级反转机会,最后导致时间不确定性。所以在实时 CORBA 中应设法采取措施,以减少 POA 中上述解复用的次数。

实时 CORBA 服务器中驻留了客户要访问的 CORBA 对象。实时 CORBA 要求 CORBA 服务器保持处于激活态,否则从客户调用时开始激活服务,时间上很难满足实时的要求。CORBA 服务器使用一个或多个 POA 来管理 CORBA 对象,当请求到达 CORBA 服务器时,服务器使用请求中的信息找到合适的 POA,并派发服务程序去处理请求。实时 CORBA 的 POA 建立在以下的策略上:

(a) 线程配置的策略 每个 POA 有一个线程池,POA 线程池的属性由该策略中所规定:即每个线程池由静态分派线程(在服务器启动时产生)和动态分配线程(在接收客户请求时产生)两部分所构成;同时 POA 线程池用两个参数(static threads, max threads)来规定静态线程和动态线程的数量。

(b) 服务器优先级的策略 线程池在向用户请求派发服务之前,以优先级巷(priority lane)形式建立原始的线程优先级,一个巷内的线程优先级相同。但在派发执行后,就由本策略来决定执行的优先级。RTCORBA 目前有两种模型:客户传递优先级和服务器声明优先级。

在前一模型中,服务方对客户请求的处理必须按照客户所要求的优先级进行,该优先级是与客户调用请求一同传递给服务方的。在后一模型中,服务方对客户请求的处理的优先级由服务器本身决定。

(c) 通信协议的策略 本策略允许服务器指定一个希望与客户联系的协议的优先级列表。此列表放在对象引用中,当客户绑定某服务后,该列表将返回客户。客户配置协议从该表中选择一协议来与服务器联系。

5) 为实时 CORBA 系统建立实时调度机制

在实时 CORBA 系统中实时应用(及其一系列活动)都是遵守端到端有时限的行为,这要依靠实时调度机制的支持来实现。实时调度机制能对应用(或其一系列活动)进行控制的依据是实时调度策略。策略的优劣将决定实时应用的时限行为。目前有静态和动态实时调度机制,调度算法有很多。

6) 实施 C/S 通信中表示层的优化

在 CORBA 系统中,客户发出的方法调用通过 stub(桩)转变为操作请求,随着后者还有操作参数等。操作和其参数被 stub 封装(Marshalling)成一体,通过下层网络设施送往服务方。在服务方它被 skeleton(框)所解封装(Demarshalling),又从请求变为方法,上调所访对象实现。当对象实现完成任务,返回结果给客户时,将再一次经历封装和解封装的过程。所以,每个请求来回双向的执行要两度经历封装-解封装,时间开销甚大,使 ORB 处理的延迟大增。为此要对封装和解封装作精心设计,使实时 CORBA 系统内 C/S 底层通信中所涉及的表示层得到优化,其中还包括动态内存和数据拷贝的优化等。

7) 实时操作系统对实时 CORBA 的支持

对于实时应用,底层操作系统的活动好像已被 ORB 所屏蔽;但实际上实时应用(及其一系列活动)的执行,是离不开操作系统的具体支持的。例如实时调度机制根据实时应用(或其一系列活动)所具有的优先级(CORBA 优先级)进行着调度,但当这一应用(或其一系列活动)进入某 ORB 端系统后,最终还得依靠当地实时操作系统所具有的本地优先级的调度能力,来实现最终的调度。所以必须解决好 CORBA 系统应用级的优先级(CORBA 优先级)与底层操作系统的优先级(本地优先级)之间存在的衔接问题,即优先级映射问题。另外,还应为实时 CORBA 系统的实现优选其实时操作系统,后者应能提供多线程且允许指定其优先级和调度策略。

8) 网络协议对实时 CORBA 的支持

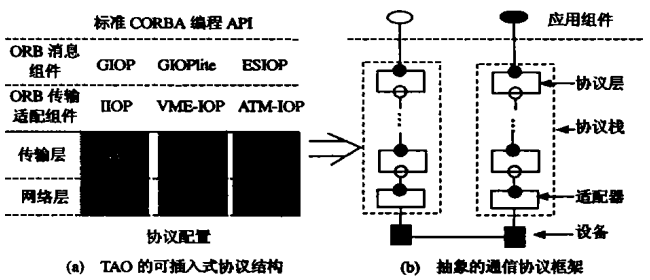

图4 支持绑定的通信协议

对于端到端的实时应用,实时网络协议的支持必不可少,尽管现在的网络实时协议已经出现不少,但是由于处于绝对主流的 TCP/IP 协议并不支持实时传输,因此以现在端到端的实时系统就应该考虑扩展网络协议的实时属性或使用插入式方法支持别的实时网络协议插入。图4(a)给出实时CORBA TAO 的可插入式协议框架。实时 CORBA TAO 为支持实时应用,需要在 VME、ATM 等支持实时的通信协议上运行,但是不能用修改上层应用的方式来适应下层通信协议的变化,因此它设计了这样一个可插入式协议框架。借助于文[12]

(下转第124页)

DCCP 流将和 TCP 流依照各自的发送速率分配带宽，而不是各自抢占带宽。这种方式极高地提高了网络资源的使用效率。随着更多方式的拥塞控制机制加入到 DCCP 协议中，拥塞控制更广泛地引入非可靠传输协议中，网络中 UDP 对网络资源的争抢造成的不公平将成为过去。

6. 以后的工作

在 DCCP 的实现方面，仍然有很多工作需要我们去完成。现在我们只是完成了 NS 中的 DCCP 的 CCID2的实现。在现有的 DCCP 的协议描述中，还有 CCID3需要实现在 NS 中。当两种实现全部完成的时候，就可以在不同的 HC 上选用不同的拥塞控制方法，测试更多的 DCCP 的性能。

DCCP 中提供的特性可以方便加入需要的特性。而新的特性的发现与加入，也需要更多的研究和探索。比如如何在 DCCP 中扩展其移动性能，如何在 DCCP 中加入安全性能。

拥有拥塞控制机制的不可靠传输协议的出现，基本上可以解决网络中不可靠流抢夺可靠流带宽的问题，只要把不可靠流使用的协议由 UDP 换成 DCCP 即可。但是，在应用不广泛、使用不方便等等情况的阻碍下，在以后一段时间，绝大多数人使用的不可靠传输协议仍然会是 UDP。因此，当不可靠流和可靠流争抢带宽的情况发生时，如何让路由器发挥出自己的作用也值得我们去研究。

结论 DCCP 为现在使用 UDP 作为传输层协议的应用提供了一种替代方案。实际上，它综合了 TCP 和 UDP 两者的优点，利用了 TCP 二十年来在拥塞控制方面的研究成果。我们的研究充分证明了这个协议在性能方面可以避免 UDP 所造成的争抢网络资源的现象，显示了 DCCP 存在的意义和它替代 UDP 成为不可靠传输层协议的潜力。

致谢 感谢陈贵海老师对本论文的指导。感谢南京大学分布式实验室在实验环境方面的支持。感谢 DCCP maillist 上对我提出问题进行中肯解答的 E. Kohler 先生。

参 考 文 献

1 Nagle J. Congestion Control in IP/TCP. RFC896, Jan. 1984
2 Jacobson V. Congestion Avoidance and Control, ACM SIGCOMM '88, Aug. 1988
3 Fall K, Floyd S. Simulation-based Comparisons of Tahoe, Reno, and SACK TCP
4 Counter Strike. http://www.counter-strike.net
5 RealAudio Player. http://www.real.com
6 Floyd S, Fall K. Promoting the Use of End-to-End Congestion Control in the Internet. IEEE/ACM Transaction on Networking, Aug. 1999
7 Kohler E, Handley M, Floyd S, Padhye J. Datagram Control Protocol. http://www.icir.org/kohler/dcp, Nov. 2001
8 Floyd S, Kohler E. Profile of Congestion Control ID 2: TCP-like Congestion Control. http://www.icir.org/kohler/dcp, May 2002
9 Padhye J, Floyd S, Kohler E. Profile of Congestion Control ID 3: TFRC Congestion Control. http://www.icir.org/kohler/dcp, June 2002
10 Floyd S, Handley M, Kohler E. Problemtatement for DCP. http://www.icir.org/kohler/dcp, Aug. 2002
11 Schulzrinne H, Casner S, Frederick R, Jacobson V. RTP: A Transport Protocol for Real-Time Applications. RFC1889
12 Stewart R, Xie Q, Morneault K, et al. Stream Control Transmission Protocol. RFC2960
13 Stevens W. TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms. RFC2001, Jan. 1997
14 Braden B, et al. Recommendations on Queue Management and Congestion Avoidance in the Internet. RFC2309, April 1998
15 Floyd S. A Report on Some Recent Developments in TCP Congestion Control. IEEE Communications Magazine, April 2001
16 Floyd S. Congestion Control Principles. RFC 2914, Sep. 2000
17 Balakrishnan H, Rahul H, Seshan S. An Integrated Congestion Management Architecture for Internet Hosts, ACM SIGCOMM, Sept. 1999
18 NS-2 simulator. http://www.isi.edu/nsnam/ns

（上接第80页）

中对协议模型的描述，可以将 TAO 的可插入式协议框架进行抽象和推广，得出图4(b)所示的通信协议模型。

小结 本文系统而详细地阐明了开放系统中实时 CORBA 技术的规范、实时 CORBA 的关键技术，为使用实时中间件实现开放系统的实时扩展提供了切实可行的方法；同时文中也启示我们两点：一、由于存在实时 CORBA1.0规范，当今研发出的可实用的实时 CORBA 系统大多使用优先级静态调度的机制，这与更为理想的动态调度尚有较大的差距。二、实时 CORBA 技术的实现不能脱离实时操作系统和实时网络的支持，但现有的实时网络协议（尤其是 TCP/IP 协议上的实时）的研究和实现与理想目标尚远，所以，要真正依靠实时中间件来构造实时开放系统，还有较多的研究工作要做。以上这些研究正是我们应该继续努力的方向。

参 考 文 献

1 骆志刚，胡健，刘锦德. 开放系统中的实时性. 计算机科学，2001(1)
2 Fay-Wolfe V, et al., Real -Time CORBA. IEEE Trans. on Parallel and Distributed Systems, 2000, 11(10)
3 刘锦德. 对于开放系统内涵的澄清. 计算机应用，1997. 6
4 Zhong Deng, Liu J W-S, Zhang L, Seri M, Frei A. An Open Environment for Real-Time Applications. Real-Time Systems Journal, 1998
5 Schmidt D C. An Overview of the Real-time CORBA Specification. IEEE Computer special issue on Object-Orient Real-Time Distributed Computing, 2000, 33(6)
6 OMG. Real- Time CORBA1.0 Adopted Specification, 1999
7 Pyarali I, et al. Applying Optimization Principle Patterns to Real-Time ORBs. the 5th USENIX COOTS'99, 1999. 5
8 OMG. The Common Object Request Broker: Architecture and Specification, OMG Document, 2.4.2, 2000.5
9 Schmidt D C. A High-performance Endsystem Architecture for Real-Time CORBA. IEEE Communications Magazine, 1997, 14(2)
10 DARPA. The Quorum Program. http: // www.darpa.mil/ito/research/quorum/ index.html. 1999
11 O'Ryan C, et al. The Design and Performance of a Pluggable Protocols Framework for Real-time Distributed Object Computing Middleware. In: Proc. of IFIP/ACM Middleware 2000 Conference, Pallisades, New York, April 2000
12 Crane J. Dynamic Binding for Distributed Systems: [Ph. D thesis]. University of London, May 1997