中间件消息队列的建模与实现

孙丽君[1]， 阮竞兰[2]， 张美忠[3]

(1. 西北工业大学 航海工程学院，陕西 西安 710072;2. 郑州工程学院 机电工程系，河南 郑州 450052;3. 西北工业大学 材料科学与工程学院，陕西 西安 710072)

摘 要:论述了面向消息中间件的建模方法,并简要讨论了该模型在Client/Server环境中的具体实现。实验证明,这种模型的实现可以弥补DEC MessageQ在动态资源管理上的不足。

关键词:中间件；消息队；异步通讯；分布式计算环境

中图分类号:TP393 **文献标识码**:A **文章编号**:1009-3516(2003)04-0092-03

随着网络体系结构的日益膨胀以及通讯协议的日趋繁杂,商业应用系统的开发面临着越来越严峻的问题:如何通过网络和通讯技术,将分布的、异质的系统有机地组织在一起,以实现资源的合理共享和信息格式的透明转换。"单一系统映象"的提出刺激了中间件(Middleware)技术的发展。中间件是位于操作系统与应用程序之间的软件,为异质环境下计算机之间信息交换提供了适当的平滑处理。因此,研究中间件模型,对于促进分布式计算环境的研制与开发具有重大意义。

1 消息队列模型的建立

消息队列是面向消息中间件(MOM)的一种异步通讯模型。利用异步通讯的非阻断性,即进程可以继续运行而不必被迫等待某个事件发生,可以并发地处理多个请求,提高系统的运行效率。因此,建模宜采用消息队列方式。实现消息队列的异步通讯有两种方法。一种是运用polling机制定时询问队列,一旦消息到达,就触发相应的处理。另一种方法是使用回调机制(call-back mechanism)。当某一事件发生时如收到一个消息,由回调机制自动通知程序。Windows操作系统就是采用这种方法。

消息队列中间件有三种类型:非持久型、持久型和事务型。不同类型的消息队列模型的建立方法也不相同。非持久型消息队列是内存型队列(Memory-based queues),一旦出现运行错误或操作失败,队列中的消息将丢失。这时,就必须重新生成消息并按原来的顺序对消息进行排队。持久型消息队列与非持久型消息队列的唯一不同在于,持久型消息队列是磁盘队列(Disk-based)而不是内存队列,因此,在某些失败的情况下,可以恢复队列中的消息。

当一个应用程序想向另一应用程序发送数据时,就发出一个PUT Q2命令。队列管理器经判断确定消息要到达的队列是本地的还是远程的。如果是远程队列,管理器就暂时将消息存储在一个本地的传送队列中,并于对应的目标平台进行协商,发起一个与远程队列管理器的对话。协商成功后,向本地传送队列提交一个GET命令,并将获取的消息通过网络发出。远程队列管理器则提交一个PUT命令,将数据存入远程队列中。远程队列在适当的时候向队列管理器发出GET Q2命令,获取发来的数据。

2 模型的实现

为了验证上节所提出的消息队列模型,作者设计了一个简单的系统。该系统由客户端和服务器端两部

收稿日期:2002-12-16
基金项目:河南省重点科技攻关项目(991140156)
作者简介:孙丽君(1968-),女,河南中牟人,讲师,博士生,主要从事计算机应用研究.

分组成,其中客户端包括客户进程和客户端中间件两个模块。服务器端包括服务器进程和服务器端中间件两个模块。下面就该系统的具体实现进行简单的论述。

2.1 客户端的实现

具体地讲,客户端由客户进程和客户端中间件组成,其中:

客户进程。负责数据、信息的组织和私有数据的打包,通过调用客户端中间件的接口原语,与服务器进行透明交互。中间件接口原语的具体实现对客户进程透明。

客户端中间件。在 Windows 下以动态联接库的形式,在 DOS 下以函数库的形式提供给用户。它的内部包括对接口原语的实现,并在消息头加上客户站点信息和设置消息的优先权标志等。然后通过 TCP/IP 的开发包与网络的下层进行连接。本系统建立在传输层之上,低层协议对中间件透明。

客户端中间件实现了 4 个接口原语:Send(发送消息)、Receive(接收消息)、Bind(建立连接)、Unbind(解除连接)。客户进程调用这些接口原语进行消息的发送和接收。

2.2 服务器端的实现

本模型在服务器端建立两个队列,一个用于接收客户发来的消息(Receive 队列),一个用于向客户发送消息(Send 队列)。服务器端的中间件部分负责通过 TLI 接口接收从 LAN 上发来的消息,经过消息的解包和处理,将消息送入 Receive 队列。消息在队列中按照到达的先后次序排列,但对消息的处理由其优先权决定。本系统所采用的消息格式见图 1。其中,客户标识代表客户机的序号,它的具体位数由系统的规模决定。假设客户机的数目为 N,则 Client_id 所需的位数不小于$[\log_2 N+1]$。优先权标识由系统定义的优先级的多少决定。若系统定义了 M 个消息优先级,则 Priority 的位数应不小于$[\log_2 M+1]$。Type 表示消息的类型。若系统有 K 类消息,则 Type 的位数应不小于$[\log_2 K+1]$。为简化起见,本系统的客户标识、优先权标识和类型域的位数分别为 4、1、2。

Client_id	Priority	Type	Data

图 1　消息格式

为了提高客户端的响应速度,充分利用服务器平台的并发性,系统为每个客户维护一个子程序,对共享队列中的消息进行处理。综上所述,服务器端对接收到客户发来的消息处理流程见图 2。

为了保证整个系统的运行效率,可以通过服务器端中间件所提供的接口原语,将消息的类型定义为:M_bind、M_send、M_receive、M_unbind。这样一来,服务器的队列控制算法将可以根据消息的类型做不同处理。控制算法的具体实现如下:

```
Run_bind( )/ * 建立连接并进入消息循环 * /
{ ……
switch(fork( ))
{ case -1:
perror("fork error!");
 exit;
default· / * parent * /
/ * 返回继续处理别的客户的连接申请 * /
return;
case 0: / * child * /
/ * 向 send 队列中发送一个连接成功的消息 * /
send_OK_message( );
while(1)
{/ * 从队列中取消息 * /
get_message_from_receive_queue( );
/ * 根据消息的类型,分别做不同处理,如果是解除连接请求的消息,则终止子进程,结束消息循环。 * /
server_process( );
send_result_message_to_send_queue( );/ * 返回响应 * /
```

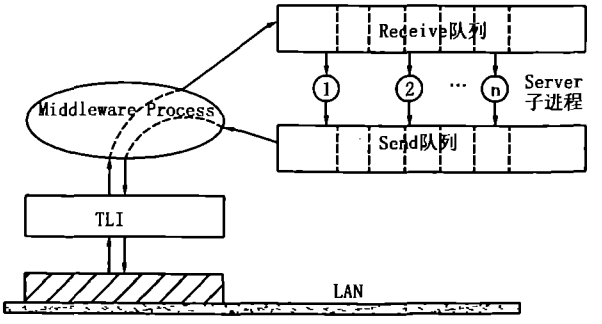

图 2　服务器端的消息处理流程

```
}
..... }
```

为了实现消息队列的多路复用,当服务器端接收到客户发送来的连接申请消息后,根据消息中的客户标识 client_id 产生一个子进程,该进程专门处理来自这一特定客户的消息。连接建立后,子进程就进入一个消息循环,一直等待着属于该客户的消息到达队列。当队列中有多个消息排队时,首先判断并处理高优先权消息,处理完所有的优先权消息后再处理普通消息。通常将消息 M_unbind 赋予一个最高的优先权,这样,客户一发出解除连接的请求,服务器处理子进程就立即结束,不再处理队列中有关该客户的剩余消息。可以设计一个消息队列监控程序,定时清理遗留在队列中不能被处理的"老死"消息。也可以采用对 M_unbind 消息赋予最低优先权的方法,这样队列中就不会存在"老死"消息,但在网络拥挤的情况下,会降低客户解除连接的响应性能。

3 结束语

本系统实现了 Client/Server 的所有组件,主要工作集中于中间件部分,旨在探索面向消息中间件的开发技术,有利于对中间件技术的进一步研究。用这种方法能够很容易地实现消息队列的动态资源管理、数据的多重发送以及消息的路由选择等功能。

参考文献:

[1] Mark Astley, Daniel C Sturman, Gul Agha. Customizable middleware for modular distributed software[J]. Communications of the ACM, 2001,44(5):99 - 107.

[2] Sameh Fakhouri, William F Jerome, Vijay K Naik, et al. Active Middleware Services in a Decision Support System for Managing Highly Available Distributed Resources[A]. IFIP/ACM International onference on Distributed Systems Platforms. 2000, New York,USA, 349 - 371.

[3] Marco Lohse, Michael Repplinger, Philipp Slusallek. An Open Middleware Architecture for Network - Integrated Multimedia [A]. Joint International Workshops on Interactive Distributed Multimedia Systems and Protocols for Multimedia Systems,2002, Coimbra, Portugal, 327 - 338.

[4] 朱三元,白英彩. 网络通信软件设计指南[M]. 北京: 清华大学出版社,1994.

[5] 胡希明,张建平. UNIX 结构分析(核心代码的结构和算法)[M]. 杭州:浙江大学出版社,1990.

(编辑:姚树峰)

Modeling and Implementation of Message Queuing Middleware

SUN Li - jun[1], RUAN Jing - lan[2], ZHANG Mei - zhong[3]

(1. College of Marine Engineering, Northwestern Polytechnic University, Xi'an, Shaanxi 710072, China; 2. Zhengzhou Institute of Technology, Zhengzhou, Henan 450052, China; 3. College of Material Science and Engineering, Northwestern Polytechnic University, Xi'an, Shaanxi 710072, China)

Abstract: A modeling method of message oriented middleware is described, and the implementation of the model under client / serve environment is introduced at the same time. Practice proves that the realization of this model can make up for the shortage of DEC Message Q on dynamic resource management.

Key words: middleware; Message Queue; asynchronism communication; distributed computing environment