开放系统中一种端到端QoS的实现机制*)

A End-to-End QoS mechanism in Open System

黎忠文　熊光泽　刘锦德

（电子科技大学计算机学院　成都610054）

Abstract　A QoS mechanism used in open systems is presented in the paper. And a series of abstract structures, describing characteristics of applications and system, as well as algorithms supporting the realization of QoS, are designed in it. On the one hand, this mechanism can address common problems found in open systems supporting QoS guarantees, such as lack of harmony QoS expressiveness, ect. On the other hand, this mechanism can support QoS dynamic degradation in order to let as many as possible applications can get system service at the sametime

Keywords　Open system, Quality of Service, Dynamic QoS degradation, Service platform for user, Service platform for resource

1　引言

开放系统是当前计算环境发展的一个重要方向，其上的许多应用都有服务质量(QoS)的需求，比如视频会议、媒体广播、IP电话、远程教育及实时控制等。QoS是目前开放系统亟待解决的问题之一，同时也是下一代网络的核心技术之一，是当前研究与开发的热点[1]。

QoS具有端对端的特性。上述应用往往要跨越系统的多个组件，甚至是多个平台，然而对用户而言，他们关心的是最终结果，而不是中间的具体细节。用户所要做的就是把QoS需求(即用户对应用的某些功能的执行速度、声频和视频的稳定性、同步性以及信息的精确度等特殊要求)，以参数的形式(常称QoS参数)提供给为之服务的系统后，等待结果。因此开放系统中，要实现QoS支持，必须要解决应用层、资源层及中间件层的相关技术。目前对QoS的研究大多属于非端对端的范围[2~5]，即在系统的局部范围(比如特定的ATM平台、CORBA中间件及多媒体应用)内讨论QoS。从表面上看，把这些在应用层、资源层和中间件层所做的研究工作组织起来就足以提供端到端的QoS支持。遗憾的是，由于它们所使用的定义、协议及平台的不一样，组织工作难以进行。即使是端对端QoS讨论，也是各有侧重，没有对开放系统中支持QoS所需解决的关键技术进行统一研究[6~8]。比如文[6]的讨论是围绕ATM平台、多媒体应用进行的，没有考虑系统的异构性及多应用性。文[7]和[8]提出了一种跨平台QoS的框架，但缺乏内在实施机制。本文在此基础上，设计了一种端对端QoS的机制以综合解决上述问题，并能很好地实现QoS的动态重协商。

2　关键技术

图1　开放系统与QoS

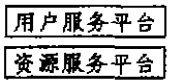

图2　QoS机制的外部视图

QoS在开放系统中所处的位置如图1所示。在开放系统中支持QoS服务，必须要解决的基本关键技术有：

1)QoS参数的一致性问题　用户和系统好比两个语言不通的人，他们对QoS参数的表达方式互不相同。比如多媒体应用中，用户对图像质量的要求可能是以每秒帧数、压缩比等方式表达，而系统的QoS语言

*)本项目获国防技术预研项目支助。黎忠文　博士生，主要研究领域为多机系统的可靠性，安全性及QoS。熊光泽　教授，博士导师，主要研究领域为实时计算机系统及其软件开发支持。刘锦德　博士生导师，主要研究领域为开放系统，分布式实时系统。

就要具体到网络的速率、压缩与解压率等。如何让两者沟通，让系统理解用户的要求并按要求组织资源为用户提供服务，就是QoS参数的一致性问题。

2）应用的抽象问题　开放系统能同时支持多种类型的应用。如果系统能为不同的应用单独提供一套QoS服务支持当然最好，但是这样将会极大地降低系统的效率，是不现实的。要达到尽可能重用系统的QoS服务的目标，就必然要解决应用的抽象问题。

3）资源的管理问题　开放系统常常是多平台的，拥有多种资源，且资源间关系复杂。要满足资源间QoS参数的一致性以及QoS参数为基础的资源分配，资源的管理机制是必不可少的。

表1 通用应用级 QoS 参数

参数名称	指　标	描　述
DataUnitSize	Avg, Min, Max	适于应用的数据单元的长度(比特)
DataUnitRate	Avg, Min, Max	适于应用的数据单元的速率(单元/秒)
EndToEndDelay	Avg, Min, Max	数据生成与使用之间的时间间隔
ErrorRatio	Avg, Max	可接受的错误率
Guarantee	—	服务保证方式
Cost	Avg, Max	费用
SecurityLevel	—	数据传输中采用的安全机制

表2 通用系统级 QoS 参数

参数名称	指　标	描　述
Net Bandwidth	Avg, Min, Max	网络带宽
Net Delay and OpSys Delay	Avg, Min, Max	网络、操作系统延迟
Net ErrorRatio	Avg, Max	可接受的传输错误率
Net Guarantee and OpSys Guarantee	— Avg, Max	网络、操作系统提供的服务保证方式
Net Cost and OpSys Cost	—	网络、操作系统的费用
Net SecurityLevel		网络采用的安全机制

表3　表1和表2之间的翻译规则

At Application Level	At System Level
App DataUnitSize×App DataUnitRate	Net. Bandwidth
App EndToEndDelay	Net. Delay +OpSys. Delay
App ErrorRatio	Net. ErrorRatio
App Guarantee	Net. Guarantee +OpSys Guarantee
App cost	Net Cost+OpSys Cost
App SecurityLevel	Net. SecurityLevel

此外，对于性能要求较高的系统还应该提供应用间QoS的动态重协商技术(QoS动态降级)，即系统资源不足时，在满足用户最小QoS需求的前题下，不需用户参预，系统自动调整各应用的资源分配，为尽可能多的用户提供服务。

为此设计了QoS机制的外部视图，如图2所示。下面将具体介绍其内部机制。

3　QoS机制的实现

3.1　QoS参数的一致性

翻译工具是解决QoS参数一致性问题的有效方法。这里采用通用应用级与通用系统级[7]两类QoS参数，它们分别代表用户与系统的QoS术语，示例见表1和表2。然后在它们之间建立翻译规则，如表3。因此下文把用户级和系统级QoS参数统称为QoS参数，不再对它们进行专门区分。

3.2　用户服务平台

用户服务平台是用户与系统打交道的窗口。对用户而言，它是系统提供的友好界面；对系统而言，它具有对应用进行抽象，屏蔽细节的作用。其主要功能有：

1）与用户沟通　用户把应用及相应的QoS参数提交给用户服务平台后，双方进行初步的QoS协商，对不合理的QoS参数进行修改。

2）把用户级的QoS参数转化为通用系统级的参数　用户服务平台中包含有多个应用代理，它们分别管理不同类型的应用。应用代理使用表3的规则进行QoS参数的翻译工作。

3）画应用流图　应用流图 $GA=\{T,E\}$ 是一个有向的无环图，它在逻辑层上描绘应用。其中 $T=\{t_1,t_2,\cdots,t_n\}$ 代表应用的n个处理步(资源分配的最小单位[8])，图中结点代表处理步；E是结点间有向边的集合。从结点 t_i 到 t_j 的数据流用它们间的有向边 (t_i,t_j) 代表。数据流向代表了各处理步的执行次序。此外每个结点都标有向量QoS_para：

$$QoS\_para(t_i)=[Q_1,Q_2,Q_3]$$

对于 t_n，Q_1 表示用户请求系统提供应用服务时所处的资源，叫源资源，记为 r_{source}；对于其它处理步，Q_1 表示处理步所需资源的类型。Q_2，Q_3 分别代表处理步对资源 Q_1 的最大及最小QoS需求。图3为一简单video应用的应用流图。

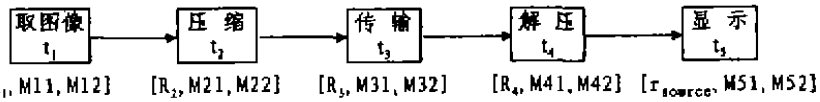

图3　video应用的应用流图

3.3　资源服务平台

资源服务平台的主要作用是屏蔽资源的物理特征，为应用分配资源，必要时它还提供应用间QoS的动态重协商，实现QoS动态降级。为此，在资源服务平台中设计了相应的抽象结构及算法。

3.3.1　抽象结构

1）资源分布图 Re-D-Graph $=\{R,E\}$ 是无向图，用来描述资源的分布情况。其中 $R=\{r_1,r_2,\cdots,r_m\}$，是系

统中资源的集合；E 是代表资源间相邻关系的边的集合。图中每个结点分别代表系统的一个资源，其上都标有向量 Res_para：

$Res\_para(r_i)=[re_1,re_2,re_3,re_4]$

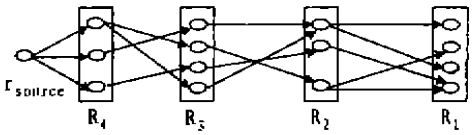

图4　图3的资源层次图

这里 re_1,re_2,re_3,re_4 分别代表 r_i 的类型、当前能提供的 QoS、可以提供的最大及最小 QoS。

2）资源层次图 Re-L-Graph。是为了方便对已分配资源的管理而设计的。当用户请求某种应用服务时，令他处在资源分布图的某个结点 r_i（某个资源）上，于是从 r_i 出发，根据该应用的应用流图 GA 和资源分布图就可画出该应用的资源层次图 Re-D-Graph＝{G，E}。G 是资源类型 R_i 的集合（图中用长方形表示），R_i 的每个分量代表一个资源（图中用小圈表示），属于同一向量的各分量属于相同的资源类，R_i 的种类及多少是由应用流图来决定。E 是资源间有向边的集合，边的方向代表应用的数据流向，边代表资源的邻接关系。注意：图上的所有资源必须能满足相应处理步的 QoS 需求。图3的资源层次图为图4所示。

3）资源分配图 Re-A-Graph。资源分配图 Re-A-Graph 记录了每一个应用的资源分配情况，便于管理应用间冲突资源。对于资源分布图 Re-D-Graph 的每条边(r_i,r_j)而言，若它的两端点 r_i 与 r_j 被分配给同一应用，则在边上标以代表该应用的标号，就成为资源分配图。

3.3.2　算法

在这些结构上，设计了 Co-D 与 Ro-D 两个资源分配算法。当系统资源完全能满足新应用的最小 QoS 需求时，用 Co-D 算法就能将资源成功地分配给应用，否则使用 Ro-D，进行应用间 QoS 的动态重协商，实现 QoS 动态降级。

1）Co-D 算法。设应用 A 请求系统服务。在 A 的资源层次图上，Co-D 寻找一条以 A 的源资源为始点，目的资源（处理步 t_i 所需的资源）为终点的路径。然后把该路径上的所有资源一一分配给 A 的各处理步。如存在多条这样的路径，则选择为 A 提供的 QoS 最大的那条。其工作过程如下：

（1）根据资源分布图和应用 A 的应用流图，画出 A 的资源层次图。

（2）对于 A 的每个目的资源，重复（3）和（4）步。

（3）使用两组临时变量 TEMP1 和 TEMP2，用 TEMP1记录 A 的资源层次图上除 r_{source} 以外的其它所有资源的 re_2；TEMP2留着记录资源层次图上各边的权值。

（4）对于 A 的资源层次图上的每一条有向边(r_i,r_j)，令 r_j 上的 TEMP1的倒数为该边的权，用 Dijsktra 算法在图上求出从源资源到目的资源的最短路径。注意当把一个资源放入最短路径时，意味作这个资源已暂时分配给应用 A，所以资源上的 TEMP1必须作相应的改变：TEMP1等于 TEMP1原有值减去暂时分配给 A 的 QoS 值。

（5）在（2）至（4）中求出的所有路径中选取最短的那条。

（6）如果以上工作成功，则按该路径把资源一一分配给 A 的各处理步并修改它们的 re_2，整个资源分配工作到此结束。否则调用 Ro-D 进行 QoS 降级。

2）Ro-D 算法。实现 QoS 协商有静态和动态两种，其中静态方法依赖用户来完成，该方法简单，但效率低；动态方法由系统自动完成，整个过程对用户透明，该方法效率高、用户满意，但设计较复杂，是目前 QoS 研究的难点之一，Ro-D 算法属于后者。

Ro-D 必须要考虑两个问题，一是多个应用中，选择哪个应用进行 QoS 降级，二是如何实现 QoS 动态降级。设当前系统中已有应用 $A_1,A_2,\cdots,A_n$。这时新应用 A 请求资源失败，激活 Ro-D 算法。不妨设应用 $A_1,A_2,\cdots,A_m(m\leqslant n)$的优先级比 A 低。令引起 A 请求资源失败的资源叫冲突资源。于是 QoS 动态降级就从它们中优先级最低的开始。其工作过程如下：

（1）把应用 $A_1,A_2,\cdots,A_m$ 按优先级从低到高依次排列；优先级相同的多个应用按冲突资源个数从少向多排列。

（2）若该队列为空，则报告用户，由用户调整它的 QoS 需求，整个 Ro-D 算法到此结束。否则取出队首应用，记为 B。根据资源分配图找到与 A 的冲突资源，为方便说明不妨设为 r_1,r_j,r_m，令它们分别属于 R_1,R_j,R_m 类资源。

（3）在资源 r_1,r_j,r_m 上暂时取消应用 B，计算它们是否能满足 A 的 QoS 需求，若失败则转（5），否则进行下一步。

（4）在 B 的资源层次图上，把 $R_i(R_1,R_m)$资源类中除了 r_1,r_j,r_m 外，最小能满足 B 的 QoS 需求的资源 r'_i（r'_1,r'_m）暂时固定分配给 B。然后在 B 的资源层次图上，寻找一条经过 r'_1,r'_j,r'_m 的最短路径。若存在多条路径，则选择使 B 的资源分配变动最小的那条。如果成功，则按该路径进行应用 B 的资源变动，并为 A 分配资源。这时 QoS 动态降级已成功实现，Ro-D 算法到

此结束。若没有可供暂时分配给 B 的资源，则转(5)步。

(5)恢复 B 的资源分配，并令 B 出队列，然后转(2)。

4 仿真实验

我们用16台由 LAN 连接的 PC 机作模拟系统，进行仿真实验。其中一台用于系统管理；一台用作用户服务平台，其上有7个 video 应用($A_1, A_2, \cdots, A_7$)。它们的应用流图和资源层次图分别为图3、图4；四台用作 R_1 类资源，模拟 video 服务器；三台用作 R_2 类资源，处理图像压缩；四台用作 R_3 类资源，负责图像传输；三台用作 R_4 类资源，处理图像解压。video 应用的各个处理步分别固定在这些资源上完成。为了简化工作，我们不妨令 $A_1, A_2, \cdots, A_7$ 对资源 R_1、R_2、R_3、R_4 的最小及最大 QoS 需求相同：M11＝5，M12＝5；M21＝5，M22＝10；M31＝5，M32＝5，M41＝5，M42＝10。下面分三种方案进行实验：

(1)每台 R_1、R_2、R_3、R_4 资源能提供的最大 QoS 量分别为15、15、10和5

(2)每台 R_1、R_2、R_3、R_4 资源能提供的最大 QoS 量分别为15、15、10和10

(3)每台 R_1、R_2、R_3、R_4 资源能提供的最大 QoS 量分别为15、15、10和15

三种方案中，系统都成功实现了 QoS 动态降级，为应用提供了满意的服务。但系统的利用率却有明显的差距，第一方案下，系统能同时为3个应用提供服务；第二、三方案分别为6和7。资源最大利用情况见图5所示。

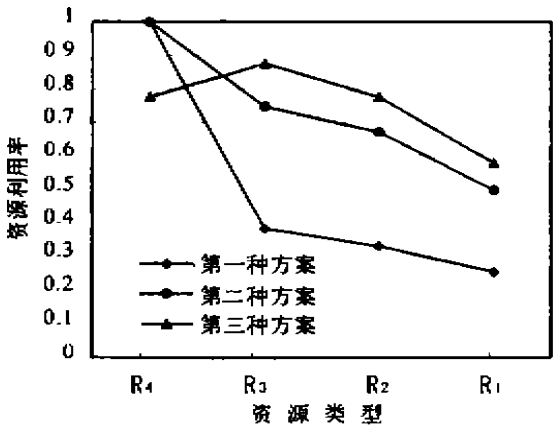

图5 系统资源利用率对比图

从图5中可以看出，在一、二方案中，资源 R_1 的使用已达最高值，虽然其它类资源还存在一定的使用空间，但系统已无法再为新应用提供服务了。这说明本算法适用于各资源搭配均匀合理的开放系统。对于有瓶颈资源的系统而言，使用本算法以提高资源利用率的效果显著。

结束语 本文对开放系统中支持 QoS 服务存在的主要问题进行了探讨，提出了一种可行的 QoS 机制。同时对 QoS 动态重协商也进行了研究，设计了相应的算法，本算法对于资源搭配均匀合理的开放系统，使用效果理想。此外，本算法适用于相邻处理步间 QoS 量相对独立的应用，对于 QoS 量间相互影响较大的应用来说，还需根据 QoS 量间具体关系，对算法进行相应的修改。这也是我们下一步的研究内容之一。

参 考 文 献

1 林闯. 多媒体信息网络 QoS 的控制 软件学报，1999，10(10)，1016～1024

2 Wang Dongxia，Dou Wenhua，Zhou Xingming. The QoS Architecture of the Survivable High-speed Multimedia Network. Journal of china institute of communications，1999，20(5)：84～88

3 Zhang Zhanjun，Liu Ye，Chen Fujie. Task Scheduling Based on QoS Control for Continuous Media Services. Journal of computer research & development，1999，36(6)：995～999

4 Dharanikota S，Maly K. QUANTA：Quality of Service Architecture for Native TCP/IP over ATM networks. HPDC'96 Proceedings；also Old Dominion University Department of Computer Science Technical report #TR-96-01，1996，5

5 Zinky J，Bakken D，Schantz R. Architectural Support for Quality of Service for CORBA Objects. Theory and Practice of Object Systems (Special Issue on OMG and CORBA)，1997

6 Aurel A L，Lim K S，Marcocini F Realizing a Foundation for Programmability of ATM Network with the Binding Architecture. IEEE Journal of Selected Areas in Communication，1996，7：1214～1227

7 Siqejra F，Cahill V. Quartz：Supporting QoS-Constrained Services in Heterogeneous Environments. 1998. Available at http://www.capes.gov.br

8 Chatterjee S，Sabata B，Sydir J. ERDOS Architecture Report. [no data]Available at http://www.erg.sri.com