ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.19, No.7, July 2008, pp.1590−1602
DOI: 10.3724/SP.J.1001.2008.01590

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

基于收益机制发布/订阅系统时间约束保障技术*

马建刚[1,2+], 黄 涛[1], 徐 罡[1], 汪锦岭[1], 叶 丹[1]

[1](中国科学院 软件研究所 软件工程技术研究开发中心,北京 100190)

[2](中国科学院 研究生院,北京 100049)

An Earning-Based Time-Constraint Assurance Technique for Distributed Publish/Subscribe System

MA Jian-Gang[1,2+], HUANG Tao[1], XU Gang[1], WANG Jin-Ling[1], YE Dan[1]

[1](Technology Center of Software Engineering, Institute of Software, The Chinese Academy of Sciences, Beijing 100190, China)

[2](Graduate University, The Chinese Academy of Sciences, Beijing 100049, China)

+ Corresponding author: E-mail: mjg@otcaix.iscas.ac.cn, http://www.iscas.ac.cn

Ma JG, Huang T, Xu G, Wang JL, Ye D. An earning-based time-constraint assurance technique for distributed publish/subscribe system. *Journal of Software*, 2008,19(7):1590−1602. http://www.jos.org.cn/1000-9825/19/1590.htm

Abstract: The publish/subscribe (pub/sub) system has a wide potential application due to its asynchronous, many-to-many and loosely-coupled communication properties. However the existing pub/sub systems can't satisfy the delay requirement of application in dynamic distributed computing environment. In order to solve this problem, this paper extends the syntax of pub/sub system, sets up the delay model, proposes an earning-based time-constraint assurance technique for distributed pub/sub system, and puts forward a scheduling algorithm named as MTEP (maximum total earning priority). The algorithm satisfies the specified delay requirement for both publisher and subscriber, makes use of available bandwidth efficiently with the price and penalty information agreed with subscribers, and adapts to the dynamic network environment. Experimental results show that the strategy can make subscribers receive many more valid events and improve system earning significantly comparing with the classical FCFS, fixed priority and least remain time priority algorithms.

Key words: publish/subscribe; event model; delay model; scheduling algorithm; priority; price; penalty

摘 要: 发布/订阅系统技术具有异步、松散耦合和多对多通信的特点,有着广阔的应用前景.但是,已有的发布/订阅系统技术不能满足动态环境下有延迟需求的应用要求.针对时间约束问题,扩展了发布/订阅系统的语法,建立了延迟模型,提出了一种基于收益机制的分布式发布/订阅系统时间约束保障技术和使系统获益最大化的调度算法MTEP(maximum total earning priority),其特点是能够满足订阅者和发布者指定延迟约束的需求,通过与订阅者商定的价格和违约成本信息来有效地利用网络带宽,适应网络环境的动态变化.实验结果表明,该调度策略和 FCFS(first

* Supported by the National Natural Science Foundation of China under Grant No.60573126 (国家自然科学基金); the National High-Tech Research and Development Plan of China under Grant Nos.2007AA04Z148, 2007AA01Z149 (国家高技术研究发展计划(863)); the National Basic Research Program of China under Grant No.2002CB312005 (国家重点基础研究发展计划(973))

Received 2006-12-04; Accepted 2007-02-05

come first service)、最短时间优先和固定优先级等传统策略相比,可使订阅者接收到的有效事件明显增多,并使系统收益显著改善.

关键词: 发布/订阅;事件模型;延迟模型;调度算法;优先级;价格;违约成本

中图法分类号: TP311 **文献标识码**: A

Internet技术的广泛应用和移动计算、网格计算以及普适计算平台的快速发展,要求分布式系统各参与者之间采用一种具有动态性和松散耦合特性的灵活的通信范型和交互机制.发布/订阅(Pub/Sub)通信范型与传统的通信范型(消息传递、RPC/RMI和共享空间)相比具有异步、多点通信的特点,使通信的参与者在空间、时间和控制流上完全解耦[1],能够很好地满足大型分布式系统松散通信的需求,因此具有良好的应用前景,已经广泛应用于银行、证券、电子商务和制造业企业信息化等各个领域.

随着网络分布式计算的发展,一些领域的应用对发布/订阅系统设施提出了新的需求,如要求能够提供适时性(timelineness)的服务质量;在一些应用场景中,事件发送到订阅者太晚就是错误的信息,如股市行情系统和环境监测系统对事件的延迟都有较严格的要求.另外,动态环境下,网络的通信带宽运行时不断进行动态改变,Pub/Sub系统设施应能适应网络环境的变化.因此,发布/订阅系统中的时间约束的服务质量保障是需要解决的重要问题.

发布/订阅系统分为集中式和分布式两种结构,其中集中式结构受中央服务器的带宽和处理能力的限制,存在单点失败和性能瓶颈,可伸缩性不好.所以,一般采用分布式事件代理网络[2].事件要通过多个事件代理才能到达订阅者,由于发布/订阅系统本身的松耦合性,其时间约束难以保证.一个事件能够满足多个订阅,而不同的订阅对相同事件的时间约束也不同;同时,它的时间延迟要受到路径上所有这些事件代理的调度决策的影响;而开放的Internet网络环境下带宽不断动态变化.已有的发布/订阅系统不能满足动态环境下有延迟需求的应用的要求,典型的Pub/Sub系统SIENA[3]以及JEDI[4]和Le Subscribe[5]等都没有考虑时间约束问题.已有的发布/订阅系统事件代理上消息的调度都采用一种先来先服务(first come first service,简称FCFS)的策略,无法满足有时间约束的紧迫性任务的需求.工业界的规范JMS[6]和CORBA通知服务规范[7]提供了设置事件的有效期和事件优先级的接口,但其方法是静态的,也不能满足不同的订阅对相同事件有不同的延迟约束的需求.在覆盖网络上提供服务质量保障(QoS assurance)来支持实时应用(如多媒体流)有大量研究,主要包括资源保留(如QRON系统[8])、优先级调度(如OverQoS[9])和QoS路由[10]的方法.文献[11]考虑了带宽和延迟参数的概率分布,采用了路径优化和延迟分解的方法来解决端到端的延迟约束问题.然而,发布/订阅系统本身具有松耦合性,发布方和订阅方之间没有明确的连接,有着多对多通信的特点,因此,传统的QoS解决方案不适用.所以,在有时间延迟约束的发布/订阅系统中改善其有效性,使尽可能多的有效消息到达仍然是一个具有挑战性的问题.

本文提出了一种基于收益机制的分布式发布/订阅系统时间约束保障技术,扩展了发布/订阅系统的语法,引入了事件的期望收益、违约成本和推迟成本等度量指标,提出了一种系统获益最大化的调度策略MTEP(maximum total earning priority),以满足应用的时间延迟约束.实验结果表明,该调度策略与网络社区经常使用的先来先服务、最短时间优先和固定优先级的策略相比,可使系统收益更大并使订阅者能够接收到更多的有效事件.

本文第 1 节给出系统的模型.第 2 节给出延迟模型和最大收益优先的调度算法.第 3 节对调度算法的性能进行实验验证和分析比较.第 4 节与国际同类工作进行比较.第 5 节总结主要研究工作和以后的工作计划.

1 时间约束的发布/订阅系统模型

系统结构分为 4 层:应用层、通知服务层、路由和调度层以及覆盖网络层,如图 1 所示.应用层是负责发布和接收事件的应用系统,有发布者和订阅者两个角色,发布者发布事件,同时指定事件的有效期;订阅者发布订阅,同时指定订阅的延迟约束和价格来表示订阅者对感兴趣的事件的延迟服务质量的需求程度,另一方面接收满足订阅要求的事件.通知服务层提供了与应用层交互的增加了时间约束参数的扩展的 Pub/Sub API,另外还包

括系统事件模型和订阅模型.路由层负责事件和订阅在各事件代理之间的转发,其中,订阅请求与价格、延迟约束一起在覆盖网络中传播,作为隐性的线索连接起发布者、中间事件代理节点和订阅者;调度算法根据事件的有效期、订阅的延迟约束、价格和违约成本以及网络带宽等信息确定事件的优先级来进行调度.覆盖网络层用于构建一个事件代理网络.

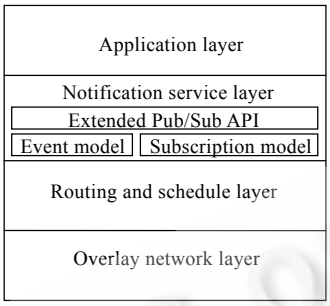


Fig.1 System structure

图 1 系统结构

1.1 事件模型和订阅模型

在发布/订阅系统中,系统的事件模型和订阅模型决定了系统的表达能力.在路由的选择过程中,要考虑事件与订阅的匹配.本文采用的路由协议和调度策略都基于该数据模型展开.下面给出它们的定义:

事件模型表示为$e=\{(a_1,v_1),(a_2,v_2),\ldots,(a_n,v_n)\}$,其中,$a_i\in A$,$A=\{a_1,a_2,\ldots,a_n\}$表示事件的属性集,$a_i$表示属性名,$v_i$表示属性$a_i$的值,$a_i$可以映射为事件属性类型$T_k$,$T_k\in\{Int,Float,Double,String,Date,Set\}$.

订阅模型表示为$S=\{(a_1,op_1,c_1),(a_2,op_2,c_2),\ldots,(a_m,op_m,c_m)\}$,其中,$F_j=(a_j,op_j,c_j)$表示属性过滤器,$c_j$是常量,$op_j\in\{<,\leq,=,\neq,>,\geq,>*,<*,\in,\subset,\subseteq\}$,其中,>*和<*表示针对*String*类型的前缀和后缀操作符,$\in,\subset,\subseteq$是集合操作符.

定义 1(事件匹配). 设有事件$e=\{(a_1,v_1),(a_2,v_2),\ldots,(a_n,v_n)\}$和订阅$S=F_1\wedge F_2\wedge\ldots\wedge F_m$,如果任给$S$的属性过滤器$F_j$,在事件$e$中存在相同的属性$a_i$,且$a_i$的值$v_i$满足$v_i\in L(F_j)$,其中,$L(F_j)=\{v|op_j(v,c_j)=\text{true}\}$且$L(F_j)\neq\varnothing$($op_j$为比较操作符),称事件$e$与订阅$S$匹配,记作$e\subseteq S$.即$e\subseteq S\Leftrightarrow\forall F_j=(a_j,op_j,c_j)\in S,\exists(a_i,v_i)\in e,a_i=a_j\wedge v_i\in L(F_j)$.

1.2 扩展的Pub/Sub API

系统通过指定服务质量参数(在这里主要是延迟约束参数),实现服务质量与事件内容分离,实现了 QoS 的解耦.应用系统通过扩展的 Pub/Sub API 来指定延迟约束,满足对适时性需求的声明.表 1 给出了 Pub/Sub 语法的扩展.

Table 1 Extended Pub/Sub syntax

表 1 Pub/Sub 语法的扩展

Pub/Sub API of existing typical system	Extended Pub/Sub system API
publish(*X*,*e*)	*publish*(*X*,*e*,*Timeout*,*timestamp*)
subscribe(*Y*,*S*)	*subscribe*(*Y*,*S*,*Deadline*,*price*,*penalty*)
unSubscribe(*Y*,*S*)	*unSubscribe*(*Y*,*S*,*Deadline*,*price*,*penalty*)
notify(*Y*,*e*)	*notify*(*Y*,*e*,*Timeout*)

发布者 *X* 在发布事件 *e* 的同时通过 *Timeout* 参数指定事件的有效期,系统自动产生时间戳(timestamp)来表明事件的发布时间.订阅者 *Y* 不但通过订阅 *S* 声明了对事件内容的过滤,而且通过 *Deadline*,*price* 和 *penalty* 指定了对事件的时间约束、获取事件的出价和系统商定的事件未按期到达的违约成本.当事件满足订阅的内容约束和时间约束时,订阅者才会被通知.我们以股票市场为例加以说明,*publish*(*X*,{*symbol*="Legend",*change*="1.5 RMB"},*Timeout*="60s",*timestamp*="Oct 8 10:30:01 MST 2006"),*subscribe*(*Y*,{*symbol*="Legend",*change*>"1 RMB"},*Deadline*="40s",*price*="1.5",*penalty*="0.15"),发布者 *X* 和订阅者 *Y* 都指定了对事件的时间约束,发布者 *X*

发布的事件和订阅者 Y 声明的订阅在内容上相匹配.系统增加了对时间约束的 QoS 功能扩展,但并没有以 QoS 合约的方式绑定参与双方,仍然保留了发布方和订阅方之间的松耦合性.

1.3 覆盖网络模型和路由

本文采用了Mesh网络作为Pub/Sub系统的拓扑结构,该覆盖网络应用在DCP[12],Gryphon[13]和XRoute[14]等典型的发布订阅系统中.在一种典型的动态消息的分发场景中,有少量的发布者和大量的订阅者,Mesh网络适合这类应用.其拓扑结构如图 2 所示,在该网络中,事件代理分为 3 类:一些事件代理连接订阅者,称为订阅边界代理;一些代理连接发布者,称为发布边界代理;另有一些代理仅作为中介转发消息,则称为中间代理.对一个事件代理而言,通过其能到达发布者的称为上游事件代理,能到达订阅者的为下游事件代理.系统采用TCP协议来支持事件代理间消息的转发.图 2 中,B_i表示事件代理,p_i表示发布客户端,s_i表示订阅客户端.

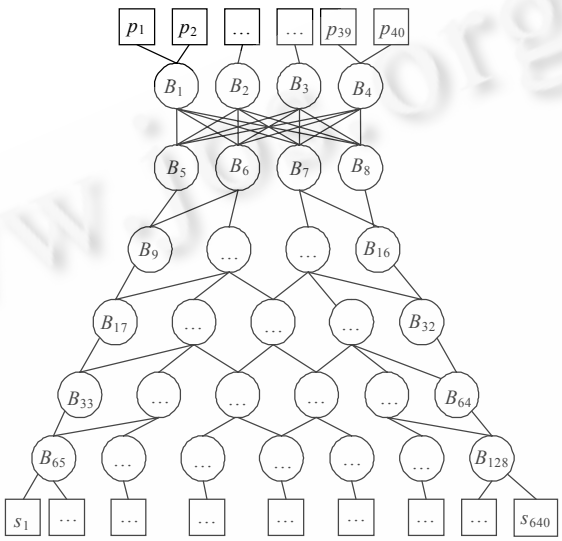


Fig.2 System topology structure

图 2 系统的拓扑结构

路由协议就是要解决如何根据事件内容,在事件代理网络中寻找一条恰当的路径,使事件低成本、高效和可靠地到达各相关的订阅者的问题.订阅要从订阅边界代理向上一步步转发到上游的发布边界代理;而事件则根据订阅一跳跳转发到下游的边界代理.本文采用的是已有的 Mesh 网络上的 Pub/Sub 系统的典型的单路径路由协议,系统仅选择一条从发布者到订阅者的路径,路径的选择标准是最小化路径上的平均传输率.本文扩展了事件转发的含义,事件转发只向与事件匹配的路由表项中的邻居代理(事件来源的代理除外)转发,且只转发到达目的订阅者的概率大于阈值的事件.该定义表示如下:

定义 2(**事件转发**). 事件转发表示为:

1) $(B_i,e_j)\in\{(B,e)|B\in V\wedge\exists(S,B)\in R_c\wedge B\neq B_{in},e\subseteq S\wedge success(e)>\varepsilon\}$;

2) $forward(B_i,e_j)$,

其中,B_i表示事件代理,e_j表示事件,$V=\{B_1,B_2,\ldots,B_n\}$表示网络中事件代理的集合,B_{in}表示事件来源的事件代理,R_c为当前事件代理的路由表,(S,B)是R_c中的路由表项,S是订阅过滤条件,B是邻居事件代理,$e\subseteq S$表示事件e与订阅过滤条件S匹配,$success(e)$表示在给定延迟约束内事件成功到达订阅者的概率,ε是一个比较小的值,是指成功到达订阅者概率的阈值.$Fwd(B_i,e_j)$是事件转发的原语,表示把事件e_j发送到事件代理B_i.

2 基于收益的调度机制

要确保发布/订阅系统的延迟时间约束,就需要建立考虑各个阶段延迟的系统延迟模型.

2.1 延迟模型

事件从发布者到订阅者要经过多个事件代理服务器,要保证其时间约束,就要考虑不同阶段的时间延迟.事件消息的延迟由从发布者到订阅者的路径上的每条连接上的延迟和在每个事件代理上的延迟构成,如图 3 所示.连接延迟又包括发布者到边界代理的延迟、相邻的事件代理之间的延迟和订阅者到边界代理的延迟.其中,事件代理上的延迟和事件代理间的延迟是主要的,而发布/订阅客户端到事件代理服务器的连接延迟一般是比较小的(移动客户端会稍长).下面分别对事件代理服务器内部延迟和事件代理间的延迟加以说明.

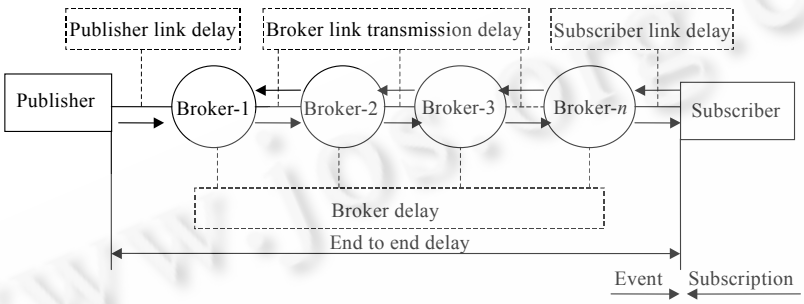


Fig.3 System end-to-end delay

图 3 系统的端到端延迟

2.1.1 事件代理上的延迟

事件代理服务器维持两种类型的消息队列:输入队列和输出队列.输入队列只有一个,用来存储到达的事件;而输出队列一般有多个,对应于该事件代理的每个下游邻居代理,用来存储经过事件匹配后等待发往该邻居代理的事件.事件代理服务器内部延迟如图 4 所示,包括输入队列的等待延迟、服务器对事件进行匹配的处理延迟和在输出队列中的等待延迟.只有当事件到达率大于事件的处理率时,输入队列的大小才大于 0.因为系统中服务器对事件进行匹配的处理是很快的[5],处理能力一般是足够的,所以,事件在输入队列的等待延迟较小,一般可忽略不计.事件服务器对每个事件进行匹配的平均处理延迟,记为PD.在输出队列中等待延迟是主要的,由于系统带宽有限,因此需要采用一种优化的调度方式来满足不同订阅的延迟需求.

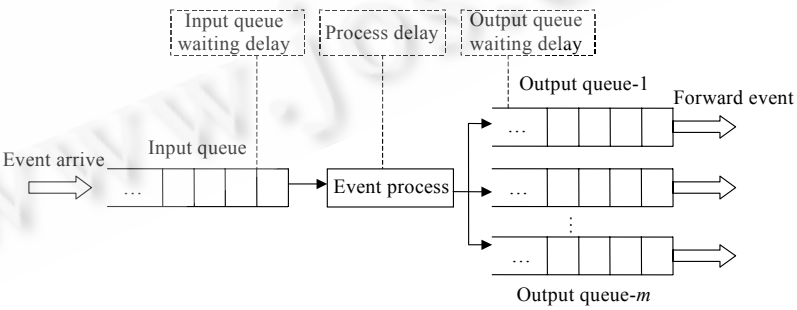


Fig.4 Delay on the broker

图 4 事件代理上的延迟

2.1.2 事件代理间的连接延迟

Pub/Sub 系统覆盖网的每条连接实际上是支撑的 Internet 网络的端到端连接.近年来的研究表明,端到端的延迟是稳定的,如文献[9]的测试数据显示,Internet 上端到端的可用带宽是相当稳定的,因此,我们可以假定

Pub/Sub 系统中的每条连接上的可用带宽满足一定的概率分布.由于 TCP 连接的传输率是由 IP 包的传输时间和 TCP 窗口的大小共同决定的,而后者又受很多因素影响.故假定 Pub/Sub 系统中的每条连接上的可用带宽满足正态概率分布.对连接 i 上的传输率定义为该连接上的单位数据传输延迟,我们使用传输 1KB 数据的所用时间来度量,记为$TR(i)$.$TR(i)$满足正态分布,记作 $TR(i)\sim N(\mu_i,\sigma_i^2)$,其中,$\mu_i$和 σ_i^2 分别表示单位数据传输延迟的数学期望和方差.每个事件代理可以通过一些网络测试工具获取到邻居代理的单位数据传输延迟的概率分布的参数.系统中不同连接上的单位数据传输延迟是独立的,于是,由多个连接构成的路径的单位数据传输延迟也满足正态分布.对有n个连接$i_1,i_2,\ldots,i_n$构成的路径r来说,令TR_r表示路径r的单位数据传输延迟,则

$$TR_r\sim N\left(\sum_{j=1}^{n}\mu_j,\sum_{j=1}^{n}\sigma_j^2\right).$$

那么,对一个大小为m个KB的事件消息,在路径r上的传输延迟为$m\times TR_r$.

2.2 系统的目标

在发布/订阅系统中,系统收益是指按期到达订阅者的事件带来的收益去除未按期到达引起的违约成本的剩余部分;订阅指定的不同的延迟需求对应不同的价格和违约成本,系统的目标是总收益最大化.假定系统有n个订阅,表示为$S_1,S_2,\ldots,S_n$.令$price(S_i)$表示订阅S_i对到达的有效事件的出价,令$penalty(S_i)$表示订阅S_i和系统商定的事件未按期到达的违约成本,令$na(S_i)$表示订阅S_i实际收到的有效事件的数目,$ni(S_i)$表示理想情况S_i应收到的事件的数目(S_i感兴趣的事件全部收到),则系统的总收益为

$$TotalEarning=\sum_{i=1}^{n}price(S_i)\times na(S_i)-\sum_{i=1}^{n}penalty(S_i)\times(ni(S_i)-na(S_i)) \tag{1}$$

系统的目标是 max($TotalEarning$).

2.3 度量指标及其计算

要使系统的总收益最大,就要考虑每个事件带来的收益,我们定义了 3 个度量指标:事件的期望收益(expected earning)、预期违约成本(expected penalty)和推迟成本(postponed cost).

2.3.1 事件的期望收益和预期违约成本

事件的期望收益是指路径上的所有剩余代理节点总是首先发送该事件,从而给系统带来的收益.对在事件代理B上的等待发送的事件e,假定它与n个订阅匹配,分别表示为$S_1,S_2,\ldots,S_n$.令$price(S_i)$表示订阅S_i对到达的有效消息的出价,$success(S_i,e)$表示在订阅S_i指定的截止期前事件e成功到达的概率,则事件e的期望收益EE_e可以通过下述公式计算得到:

$$EE_e=\sum_{i=1}^{n}price(S_i)\times success(S_i,e) \tag{2}$$

根据公式(2),事件的期望收益由对该事件感兴趣的订阅的数目、订阅的出价和该事件在截止期前成功到达目的订阅者的概率来决定.

事件的预期违约成本是指如果事件未按期到达订阅者而引起的系统需支付的违约成本.令$penalty(S_i)$表示订阅S_i和系统商定的事件未按期到达的违约成本,从而事件e的预期违约成本EP_e可以定义为

$$EP_e=\sum_{i=1}^{n}penalty(S_i)\times\big(1-success(S_i,e)\big) \tag{3}$$

下面讨论一下如何计算事件到达订阅者的成功概率$success(S_i,e)$.令事件e到达当前代理已经产生的延迟为$pt(e)$,令函数$ft(S_i,e)$为从当前代理到目的订阅S_i的延迟,r表示从当前代理到目的订阅的路径.于是,$ft(S_i,e)$由 4 部分组成:路径r上的所有节点的处理延迟、路径r上的所有节点的调度延迟、路径r上的所有代理间连接的传输延迟以及订阅者和边界代理之间的连接延迟.因为路径上剩余节点的调度延迟都是未知的,根据期望收益的定义,在剩余节点都是首先调度,可以假定剩余节点上的调度延迟为 0.因此,$ft(S_i,e)$可以表示为

$$ft(S_i,e)=nodenum\times PD+size(e)\times TR_r+SBD \tag{4}$$

其中,*nodenum*表示路径*r*上的事件代理数目(因为在Mesh网络中,每层网络节点到订阅者经过的事件代理数目为网络的总层数与该层层号的差值,故对给定的事件代理而言,*nodenum*为确定的值),*PD*表示事件代理的处理延迟,*size*(*e*)表示事件*e*的大小,TR_r指路径*r*的单位数据传输延迟,*SBD*表示订阅客户端到边界代理的连接延迟.

令订阅S_i的最小允许延迟为$dl(S_i)$,是指事件*e*的有效期和订阅S_i的截止期的较小者(一般情况下,事件*e*的有效期要大于订阅S_i的截止期),表示为公式(5):

$$dl(S_i)=\min\{timeout(e),deadline(S_i)\} \tag{5}$$

于是,事件到达订阅者的成功概率可以表示为

$$success(S_i,e)=P(pt(e)+ft(S_i,e)\le dl(S_i)) \tag{6}$$

由$TR_r\sim N(\mu,\sigma^2)$,其中,μ和σ^2分别表示路径*r*上单位数据传输延迟的数学期望和方差,令 $X=\frac{TR_i-\mu}{\sigma}$,则可以将其转化为标准正态分布 $X\sim N(0,1)$,于是由公式(4)和公式(6)可以得到:

$$success(S_i,e)=P\left(X\le\frac{dl(S_i)-pt(e)-nodenum\times PD-SBD-\mu\times size(e)}{size(e)\times\sigma}\right) \tag{7}$$

从而事件到达订阅者的成功概率可以通过查找标准正态分布函数表获得,进而事件的期望收益和预期违约成本可以通过公式(2)和公式(3)求得.

2.3.2 事件的推迟成本

事件的期望收益反映出高成功概率到达目的订阅的事件具有高优先级,但该指标不能反映事件消息的紧迫性.到达目的订阅成功概率高的事件并不一定是紧迫的,即推迟这种消息的发送可能是无害的,于是,低成功概率到达而又紧迫的事件可以提前发送.因此,引入推迟成本(*PC*)来表示消息的紧迫性.对在事件代理*B*上的等待发送的事件*e*,事件的期望收益EE_e是指路径上的所有剩余代理节点均首先发送该事件给系统带来的收益.另一种期望收益我们称为推迟期望收益,是指当前节点第 2 个发送事件*e*,其余节点仍然第 1 个发送该事件,给系统带来的收益,记为 EE'_e.相应地推迟一次,事件的预期违约成本称为推迟违约成本,记为 EP'_e.称事件推迟发送一次引起的系统收益的变化的差值为推迟成本,记为PC_e.该成本越高,发送该事件的任务就越紧迫.要计算 EE'_e,则需知道发送第 1 个事件需要的时间.因为此刻第 1 个发送的事件尚未选定,只能通过事件的平均大小和当前代理到对应邻居的连接的单位数据传输延迟的均值的乘积来估算,记为*LMT*.在当前代理第 2 个发送事件*e*,其余节点仍然第 1 个发送该事件的情形下,令函数$ft'(S_i,e)$为从当前代理到目的订阅S_i的延迟,则可以表示为

$$ft'(S_i,e)=ft(S_i,e)+LMT \tag{8}$$

令$success'(S_i,e)$表示在订阅S_i指定的截止期之前事件*e*成功到达的概率,则

$$success'(S_i,e)=P(pt(e)+LMT+ft(S_i,e)\le dl(S_i)) \tag{9}$$

于是,推迟期望收益 EE'_e 可以表示为

$$EE'_e=\sum_{i=1}^{n}price(S_i)\times success'(S_i,e) \tag{10}$$

推迟违约成本 EP'_e 则可以表示为

$$EP'_e=\sum_{i=1}^{n}penalty(S_i)\times\left(1-success'(S_i,e)\right) \tag{11}$$

那么,推迟成本PC_e可以表示为

$$PC_e=EE_e-EP_e-\left(EE'_e-EP'_e\right) \tag{12}$$

2.3.3 事件的优先级

事件的期望收益、预期违约成本和推迟成本都是决定事件发送顺序的重要因素.事件的期望收益越大,表示该事件发送给系统带来的潜在收益会较大;预期违约成本越大,表示事件如果未能按期到达,则对系统的潜在损失越大;推迟成本越高,表示事件紧迫性越高;它们共同决定了事件发送的优先级.因此,我们把事件的优先级定义为

$$priority(e)=weight\times EE_e+(1-weight)\times(PC_e+EP_e) \tag{13}$$

其中,$weight\in[0,1]$.

公式(13)中,*weight* 表示期望收益的权重,系统中具体值可以通过取经验值获得最好的调度效果.

由公式(2)和公式(13)可以看出,一个事件当它满足订阅的数目越多,事件的违约成本越大,订阅的出价越高和该事件在截止期前成功到达目的订阅者的概率越大、紧迫度越高时,其优先级越高.

2.3.4 过期事件的检测和处理

系统应该能够检测到已经到过期值(timeout)的事件和在指定延迟期间不能到达任何订阅的事件,并尽早删除,避免不必要的网络流量.为了提高可用带宽的利用率,事件代理不仅删除到达过期值的事件,而且删除即使未到事件的过期值但在指定延迟期间不能到达任何订阅的事件.为了表述方便,本文把已经到过期值的事件和在指定延迟期间不能到达任何订阅的事件称为过期事件,其定义如下:对在事件代理B上等待发送的事件e,假定它和n个订阅匹配,分别表示为$S_1,S_2,\ldots,S_n$;用布尔函数$istimeout(e)$表示事件是否到过期值;事件e称为过期事件当且仅当下述条件成立:

$$istimeout(e)\vee(\forall i\in\{1,2,\ldots,n\}:success(S_i,e)\le\varepsilon) \tag{14}$$

其中,ε是一个较小的值,表示成功到达订阅者概率的阈值.此时,系统从事件代理 B 上删除事件 e.

2.4 调度算法描述与分析

为了便于算法描述,首先给出路由表的数据结构,可以表示为如下七元组的集合:{(*subId*,*filter*,*deadline*,*price*,*penalty*,*nb*,*nodenum*)},其中,*subId* 表示订阅的标示,*filter* 表示其订阅过滤条件,*deadline* 表示订阅声明的允许消息到达的截止期,*price* 是指该订阅愿意为到达的有效消息的出价,*penalty* 是指消息未按期到达系统需要支付的违约成本,*nb* 表示当前代理的邻居代理,*nodenum* 是指从当前代理到订阅者的路径上经过的事件代理个数.

通过第 2.1 节~第 2.3 节的分析,下面给出调度算法 MTEP 的描述,其中,图 5 给出了计算事件优先级的算法伪代码描述,图 6 给出了最大收益的优先级调度算法.图 5 中,代码行[3]~[14]计算与订阅匹配成功的事件的期望收益、预期违约成本和推迟成本,代码行[15]对事件队列中的过期事件删除,代码行[16]对没有过期的事件计算其发送的优先级.

Procedure *calculatePriority*(*eq*,*nb*)
Input: *eq* is an event queue;
nb is a neighbor broker of current event broker
[1] BEGIN
[2] **foreach** $e\in eq$ **do**
[3] **foreach** $e\sqsubseteq S_i\wedge S_i\in nb.filter$ **do**
[4] $dl(S_i)=\min\{timeout(e),deadline(S_i)\}$
[5] $success(S_i,e)=P\left(X\le\dfrac{dl(S_i)-pt(e)-nodenum\times PD-SBD-\mu\times size(e)}{size(e)\times\sigma}\right)$
[6] **if** $(success(S_i,e)\ge\varepsilon)$ **then** $EE_e=EE_e+price(S_i)\times success(S_i,e)$
[7] $EP_e=EP_e+penalty(S_i)\times(1-success(S_i,e))$
[8] $success'(S_i,e)=P(pt(e)+LMT+ft(S_i,e)\le dl(S_i))$
[9] **if** $(success'(S_i,e)\ge\varepsilon)$ **then** $EE'_e=EE'_e+price(S_i)\times success'(S_i,e)$
[10] $EP'_e=EP'_e+penalty(S_i)\times(1-success'(S_i,e))$
[11] **endif**
[12] **endif**
[13] **endloop**
[14] $PC_e=EE_e-EP_e-(EE'_e-EP'_e)$
[15] **if** $(EE_e=0\vee istimeout(e))$ **then** *eq.remove*(*e*)
[16] **else** $priority(e)=weight\times EE_e+(1-weight)\times(PC_e+EP_e)$
[17] **endif**
[18] **endloop**
[19] **END**

Fig.5 Calculate the event priority

图 5 计算事件的优先级

```
Procedure MaxTotalEarningSchedudle(eq,nb)
Input: eq is an event queue;
nb is a neighbor broker of current event broker
[1] BEGIN
[2] maxPriorityEvent=null
[3] calculatePriority(eq,nb)
[4] foreach e∈eq do
[5] if e.priority>maxPriorityEvent.priority then maxPriorityEvent=e
[6] endif
[7] endloop
[8] forward(nb,maxPriorityEvent)
[9] eq.remove(maxPriorityEvent)
[10] END
```

Fig.6 Max total earning priority scheduling algorithm
图 6 最大收益的优先级调度算法

最大收益的优先级调度算法如下,其中,代码行[3]调用上面定义的计算事件优先级的函数,代码行[4]~[7]求出具有最大优先级的事件,代码行[8]和[9]表示向对应的邻居代理转发最大优先级的事件,并从事件队列中删除该事件.

MTEP 调度算法的开销包括两部分:计算事件队列中的事件优先级和选择队列中优先级最大的事件.假定事件队列中有 n 个等待发送的事件,每个事件在路由表中至多有 m 个匹配的订阅,则计算事件优先级的时间为 $O(mn)$,选择队列中优先级最大的事件的时间为 $O(n)$.故 MTEP 算法的时间复杂度为 $O(mn)$,与 FCFS 和固定优先级的调度算法相比,会给事件代理带来一些性能负担.

3 实验验证

我们用 Java 语言开发了 Pub/Sub 原型系统,并实现了调度算法 MTEP,对其进行了仿真实验,评价其在不同负载条件下的性能,探讨了算法中权重的取值对系统性能的影响.为了评价 MTEP 调度算法的性能,我们还实现了另外 3 种网络环境下的调度策略:

1) 先来先服务(简记为 FCFS)的策略,该策略对先到的消息先转发,已有的典型的 Pub/Sub 系统,如文献[3,4,10−12],均采用这种策略;

2) 固定优先级(简记为FIX(fixed priority))的调度策略,该策略对事件设置固定的优先级级别,JMS规范[6]和CORBA通知服务规范[7]采用了这种策略;

3) 最短时间优先(简记为 LRT(least remain time priority))的策略,该策略对剩余时间最短的事件优先转发,其中,事件满足订阅指定的截止期与事件已用时间的差值为剩余时间,对事件满足多个订阅的,则取剩余时间的均值来调度.

我们比较了这 4 种策略在相同环境、相同负载条件下的性能表现.

3.1 性能评价指标

实验中,我们采用下面两种性能指标来评估不同的调度策略:

1) 系统总收益,可以由公式(1)来度量;

2) 成功到达率,是指订阅实际收到的有效事件占理想情况应收到的事件的百分比.假定系统有n个订阅,表示为$S_1,S_2,\ldots,S_n$,令$na(S_i)$表示订阅S_i实际收到的有效事件的数目,$ni(S_i)$表示理想情况下S_i应收到的事件的数目,于是成功到达率为

$$\frac{\sum_{i=1}^{n} na(S_i)}{\sum_{i=1}^{n} ni(S_i)} \tag{15}$$

3.2 实验环境建立

我们模拟了一个分层的事件代理网络,如图 2 所示,该网络与文献[12,13]中使用的拓扑结构类似.在模拟网络中,有 128 个事件代理,被分为 6 层.在第 1 层网络中有 4 个事件代理,每个事件连接了 10 个发布客户端;第 2 层有 4 个事件代理,每个事件都与第 1 层的所有代理相连.第 3~6 层分别有 8,16,32 和 64 个事件代理,其中每个事件代理随机地与相邻的上层的两个代理相连.第 6 层的每个事件代理连接 10 个订阅客户端.于是,系统中共有

640 个订阅客户端和 40 个发布客户端.实验中每条连接的传输率(单位数据传输延迟)在 40ms~80ms 之间随机产生,传输率的标准差为 20ms,在事件代理上每个事件的处理延迟为 2ms,发布/订阅客户端到边界代理的连接延迟均为 20ms,实验中成功到达订阅的概率的阈值ε取为 4%.

每个发布者以一定的速率持续不断地发布事件.每个发布者平均每分钟发布的事件数称为系统的发布率.每次测试时间为两个小时.每个事件的大小为 50KB.对事件发布,指定对其感兴趣的订阅占所有订阅的比率,为匹配率,表示为mr,设所有可能的属性数量为a_n,每个订阅以及每个事件中的属性数量均为a_m,则

$$mr=(1/C_{a_n}^{a_m})\times(1/2)^{a_m}.$$

实验中,a_n和a_m均为 4,则匹配率为 6.25%.在仿真实验中,随机产生出各事件和过滤条件,以使它们满足给定的匹配率要求.每个事件的内容是 4 个"属性=值"的集合,表示为$\{(a_1,x_1),(a_2,x_2),(a_3,x_3),(a_4,x_4)\}$;而每个订阅是 4 个"属性<值"的集合,表示为$\{(a_1,<,c_1),(a_2,<,c_2),(a_3,<,c_3),(a_4,<,c_4)\}$,事件和订阅的属性的值为Double型,在0~10之间随机产生.系统中每个订阅的延迟从集合{20s,24s,30s,40s,60s}中随机产生,其相应的价格和违约成本为集合{3,2.5,2,1.5,1}和{0.3,0.25,0.2,0.15,0.1}中的对应元素.对固定优先级的策略,事件的优先级分为 5 个等级,从集合{1,2,3,4,5}中随机产生.

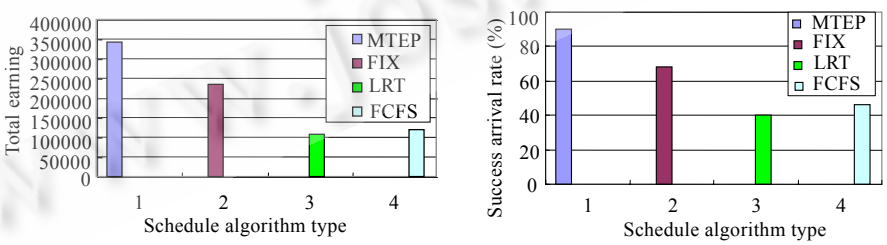
测试平台为 SUN 工作站,型号为 Sun-Blade-1000,CPU 为 SPARC,1024M 内存,操作系统为 Solaris(版本为 SunOS Release 5.8),Java 虚拟机为 JDK1.4.2.

3.3 实验结果与分析

3.3.1 算法性能比较

从图 7 和图 8 可以看出 MTEP 算法的优势明显,有较高的成功到达率和系统总收益.图 7 比较了不同算法在发布率取 1 时的总收益和成功到达率,MTEP 算法的权重取值为 0.4.MTEP 算法的总收益分别是 FIX,LRT 和 FCFS 算法的 1.47,3.16 和 2.84 倍,成功到达率分别是 FIX,LRT 和 FCFS 算法的 1.33,2.26 和 1.97 倍.

图 8 比较了不同算法在发布率变化时的总收益和成功到达率.当负载很轻时,这 4 种算法的成功到达率都很高,系统收益也大致相同;但是,随着负载的增加,FCFS,LRT 和 FIX 算法性能下降得很快,而 MTEP 算法还能保持较好的性能.这是因为 FCFS 对事件不加区分,造成高负载情况下事件延误率很高;LRT 能够反映出对剩余时间短的事件的偏好,但是剩余时间短的事件在高负载情况下,可能没有到达订阅者就已经过期,其成功到达订阅者的可能性比较小;FIX 策略反映出优先级高的事件的偏好,但是事件优先级是固定的,没有考虑网络带宽等因素;MTEP 算法考虑了订阅的延迟约束、事件的有效期和网络带宽等因素,采用动态优先级策略,适应了网络环境变化.另外,MTEP 算法随发布率的增加,成功到达率在下降,而系统的收益却在显著增加.这是因为 MTEP 算法对订阅出价高和高概率到达订阅者的事件优先转发,随着发布率的提高,实际到达订阅者的有效事件数目增加.

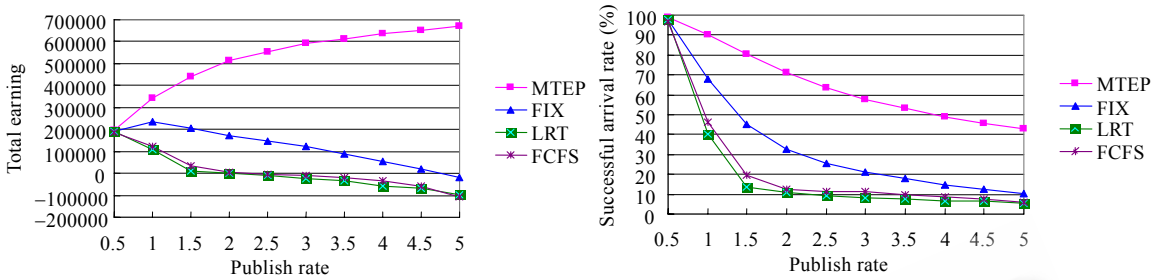


(a) Comparison on system total earning (b) Comparison on successful arrival rate

(a) 不同算法的系统总收益比较 (b) 不同算法的系统成功到达率比较

Fig.7 Performance comparison of different algorithms

图 7 不同算法的性能比较



(a) Total earning vs. publish rate
(a) 不同算法的系统总收益随负载的变化

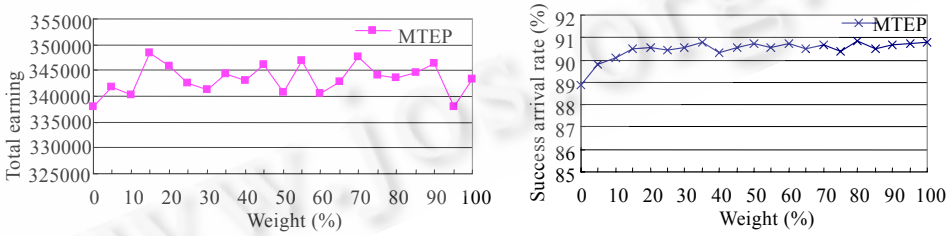
(b) Success arrival rate vs. publish rate
(b) 不同算法的系统成功到达率随负载的变化

Fig.8 Performance comparison of different algorithms under the various load

图 8 不同算法随负载变化的性能比较

3.3.2 权重参数分析

在第 2.3.3 节中,事件的优先级中期望收益的权重并没有选定,它的取值会显著影响调度算法的性能.图 9 中改变 MTEP 算法的权重(weight)的取值,从 0%~100%,测试了系统的总收益和成功到达率.其中成功到达率在 88.85%和 90.84%之间,可以看出,MTEP 算法使系统维持在一个较高的成功到达率.系统的总收益在 332 927.9 和 348 287.2 之间,可以看出,MTEP 算法使系统维持在一个较高的总收益上.当 *weight*→0 时,系统主要以事件的成本(推迟成本和违约成本)来进行调度,推迟成本越高的事件,其紧迫性越高,违约成本越大,则事件未按期到达带来的潜在损失越大,紧迫性高的和违约成本高的事件被优先调度;当 *weight*→100%时,系统完全以事件的期望收益来进行调度,期望收益越大的事件越被首先调度;当 *weight* 取中间值时,体现了对事件的期望收益、违约成本和紧迫性的平衡.当 *weight*>20%时,系统的总收益和成功到达率趋于稳定,故可以看出,事件的期望收益是系统性能的主要影响因素,而推迟成本的影响相比较小.当权重为 80%时,成功到达率取得最大值 90.84%;而当权重为 15%时,系统收益达到最大值 348 287.2.可以看出,当权重取值使系统总收益最大时,成功到达率并没有达到最大,只是取得接近最大值的值.这是因为当系统收益最大时,调度算法尽可能满足能给系统带来较大收益的事件的发送,导致部分收益不大但却容易到达的事件发送延迟.另外,由实验数据计算结果可知,系统总收益和成功到达率的相关系数为 0.403,从而两者的相互联系是正相关的.因此,系统可以根据目标选择最优的权重取值来优化调度算法的性能.



(a) Weight vs. system total earning
(a) 权重对系统收益的影响

(b) Weight vs. successful arrival rate
(b) 权重对系统成功到达率的影响

Fig.9 Influence of weight on performance of MTEP algorithm

图 9 权重对 MTEP 算法性能的影响

4 相关工作比较

目前,典型的Pub/Sub系统如SIENA[3],JEDI[4],Le Subscribe[5]和Gryphon[13]等都没有考虑时间约束问题.近年来出现了发布/订阅系统的服务质量保障方面的研究[15−19].文献[15]提出了发布/订阅系统需要解决的延迟约束、带宽、消息优先级和发送顺序等问题,但没有针对延迟约束的服务质量问题给出具体的解决方法.文献[16]提出了一种支持服务质量的事件代理结构.IndiQos[17]扩充了Hermes系统,并把服务质量需求表达为订阅和广

告的附加属性,允许订阅者指定最大延迟,采用了资源保留的机制来解决路由问题,但是不同的订阅对同一个事件有不同的延迟需求,会导致资源消耗过大,不适合大规模分布式计算环境.文献[18]提出在发布/订阅系统中使用价格机制来解决过载问题.文献[19]提出了Pub/Sub系统的服务质量约束的一组参数和路由的一般规则,但却没有针对端到端的延迟约束提出有效的方法.本文利用调度策略,提出了一种基于收益机制的分布式Pub/Sub系统的延迟约束的有效解决方法.

工业界的规范JMS[6]和CORBA通知服务规范[7]都定义了消息分发的服务质量选项,提供了设置事件的有效期和事件优先级的接口,但其方法是静态的,而且没有像MTEP这样提供支持订阅指定时间约束的方法.OMG的DDS(data distribution service)规范[20]定义了对时间约束的截止期的服务质量策略和延迟目标策略.本文提出的MTEP算法采用了动态优先级策略,并考虑了网络带宽等因素,适应了网络环境变化.

已有的覆盖网络上的服务质量的研究工作[8−11]主要采用传统的单播和多播通信方式,与Pub/Sub系统的多对多的间接通信方式不同;传统的服务质量保障的研究都是基于保留着必要资源的建立的连接或渠道来实现服务质量保障(如延迟等),而发布/订阅系统本身具有松耦合性,发布方和订阅方之间没有明确的连接,因此,传统的服务质量解决方案不适用.

5 结 论

分布式发布/订阅系统中的时间约束的服务质量保障是需要解决的重要问题.本文针对时间约束问题,扩展了发布/订阅系统的语法,建立了延迟模型,提出了一种基于收益机制的分布式发布/订阅系统时间约束保障技术和使系统获益最大化的调度算法 MTEP,其特点是能够满足事件和订阅指定延迟约束的需求,充分利用了网络带宽,适应网络环境的动态变化.仿真实验结果表明,该调度策略与网络社区经常使用的 FCFS、最短时间优先和固定优先级的策略相比,能够显著改善系统收益并使订阅者接收到更多的有效消息,较好地满足应用的延迟约束需求.下一步我们准备扩展 MTEP 调度算法,应用在其他类型的事件代理网络上以及探索其他价格模型对算法性能的影响.

References:

[1] Eugster PT, Felber PA, Guerraoui R, Kermarrec AM. The many faces of publish/subscribe. ACM Computing Surveys, 2003,35(2): 114−131.

[2] Ma JG, Huang T, Wang JL, Xu G, Ye D. Underlying techniques for large-scale distributed computing oriented publish/subscribe system. Journal of Software, 2006,17(1):134−147 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/17/134.htm

[3] Carzaniga A, Rosenblum DS, Wolf AL. Design and evaluation of a wide-area event notification service. ACM Trans. on Computer Systems, 2001,19(3):332−383.

[4] Cugola G, Nitto ED, Fuggetta A. The JEDI event-based infrastructure and its application to the development of the OPSS WFMS. IEEE Trans. on Software Engineering, 2001,27(9):827−850.

[5] Pereira J, Fabret F, Llirbat F, Shasha D. Efficient matching for Web-based publish/subscribe systems. In: Etzion O, Scheuermann P, eds. Proc. of the 7th Cooperative Information Systems. LNCS 1901, Eilat: Springer-Verlag, 2000. 162−173.

[6] Sun Microsystems Inc. JMS specification version 1.1. 2002. http://java.sun.com/products/jms

[7] OMG. CORBA notification service specification version 1.0.1. 2002. http://www.omg.org/corba

[8] Li Z, Mohapatra P. QRON: QoS-Aware routing in overlay networks. IEEE Journal on Selected Areas in Communications, 2004, 22(1):29−40.

[9] Subramanian L, Stoica I, Balakrishnan H, Katz R. OverQoS: An overlay based architecture for enhancing Internet QoS. In: Proc. of the USENIX 1st Symp. on Networked System Design and Implementation. San Francisco: USENIX Press, 2004. 71−84. http://nms.lcs.mit.edu/papers/overqos-nsdi04.pdf

[10] Cui Y, Wu JP, Xu K, Xu MW. Research on Internetwork QoS routing algorithms: A survey. Journal of Software, 2002,13(11): 2065−2075 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/13/2065.pdf

[11] Lorenz DH, Orda A. QoS routing in networks with uncertain parameters. IEEE/ACM Trans. on Networking, 1998,6(6):768−778.

[12] Snoeren AC, Conley K, Gifford DK. Mesh-Based content routing using XML. In: Marzullo K, ed. Proc. of the 18th ACM Symp. on Operating Systems Principles. New York: ACM Press, 2001. 160−173.

[13] Bhola S, Strom R, Bagchi S, Zhao Y, Auerbach J. Exactly-Once delivery in a content-based publish-subscribe system. In: Lala J, ed. Proc. of the Int'l Conf. on Dependable Systems and Networks. Washington: IEEE Computer Society Press, 2002. 7−16.

[14] Chand R, Felber PA. A scalable protocol for content-based routing in overlay networks. In: Avresky D, ed. Proc. of the 2nd IEEE Int'l Symp. on Network Computing and Applications. Washington: IEEE Computer Society Press, 2003. 123−130.

[15] Behnel S, Fiege L, Mühl G. On quality-of-service and publish-subscribe. In: Hinze A, Pereira J, eds. Proc. of the 26th IEEE Int'l Conf. on Distributed Computing Systems Workshops. Lisboa: ACM Press, 2006. 20−25.

[16] Araujo F, Rodrigues L. Quality of service in indirect communication systems. In: Proc. of the 4th European Research Seminar on Advances in Distributed Systems (ERSADS 2001). Bertinoro, 2001. http://www.cs.unibo.it/ersads/papers/araujo.ps

[17] Carvalho N, Araujo F, Rodrigues L. Scalable QoS-based event routing in publish-subscribe systems. In: Proc. of the 4th IEEE Int'l Symp. on Network Computing and Applications. Washington: IEEE Computer Society, 2005. 101−108. http://www.gsd.inesc-id.pt/~ler/reports/nca05.pdf

[18] Jerzak Z, Fetzer C. Handling overload in publish/subscribe systems. In: Hinze A, Pereira J, eds. Proc. of the 26th IEEE Int'l Conf. on Distributed Computing Systems Workshops. Lisboa: ACM Press, 2006. 32−37.

[19] Zieba B, van Sinderen M, Wegdam M. Quality-Constrained routing in publish/subscribe systems. In: Terzis S, Donsez D, eds. Proc. of the 3rd Int'l Workshop on Middleware for Pervasive and Ad-Hoc Computing. New York: ACM Press, 2005. 1−8.

[20] Object Management Group. Data distribution service for real-time systems specification. 2002. http://www.omg.org/technology/documents/formal/data_distribution.htm

附中文参考文献:

[2] 马建刚,黄涛,汪锦岭,徐罡,叶丹.面向大规模分布式计算发布订阅系统核心技术.软件学报,2006,17(1):134−147. http://www.jos.org.cn/1000-9825/17/134.htm

[10] 崔勇,吴建平,徐恪,徐明伟.互联网络服务质量路由算法研究综述.软件学报,2002,13(11):2065−2075. http://www.jos.org.cn/1000-9825/13/2065.pdf

马建刚(1977－),男,河南郑州人,博士生,主要研究领域为分布式计算,中间件技术.

黄涛(1965－),男,博士,研究员,博士生导师,CCF 高级会员,主要研究领域为软件工程,分布式计算.

徐罡(1973－),男,博士,主要研究领域为分布式计算,中间件技术,企业应用集成.

汪锦岭(1974－),男,博士,主要研究领域为分布式计算,中间件技术.

叶丹(1971－),女,博士,副研究员,主要研究领域为分布式计算,中间件技术,企业应用集成.