端到端高速网络中的通用 QoS 解决方案

张　昱

(浙江大学 信息学院 信电系, 浙江 杭州 310027)

摘　要: 提出并实现了一种针对端到端高速网络模型的简单有效可行的通用 QoS 解决方案模型。利用控制数据包容量和服务器端阻塞发送线程的方式限制低级别服务的带宽, 从而保证高级别服务拥有更高带宽。根据概率原理设计了四种支持 QoS 的竞争调度算法, 并进一步探索了用于提高带宽利用率的局部搜索和全局优化的手段。基于此方案实现了一个有效的、可扩展的 QoS 控制核心, 并设计实现了一个应用实例。

关键词: 服务质量; 有效带宽; 服务带宽; 服务级别

中图法分类号: TP393　　**文献标识码:** A　　**文章编号:** 1001-3695(2006)02-0225-04

Universal QoS Solution for End-to-End High-Speed Network

ZHANG Yu

(Dept. of Information & Electronic Engineering, College of Information, Zhejiang University, Hangzhou Zhejiang 310027, *China)*

Abstract: An universal QoS solution model for the end-to-end high-speed network model has been presented and implemented, which is simple, effective and practicable. In this solution, with the data packet size controlled and the transmitting thread blocked at the server side, the bandwidth of the lower level service will be restricted, and higher bandwidth can be guaranteed for the higher level service. Four competition scheduling algorithms have been presented based on the probability principles. Then the local search and global optimization methods to increase the bandwidth utilization has been studied. Based on this solution, an effective and scalable QoS controlling kernel has been implemented, and then an application example has been implemented.

Key words: QoS; Effective Bandwidth; Service Bandwidth; QoS Level

QoS 技术的核心问题是如何在只具有“尽力而为”(Best-effort)服务特性的电信网络等通信网络中提供超越这一特性的服务质量, 为保障复杂网络中重要程度高的数据更有效地传输创造条件[1]。目前, 规模越来越大、种类越来越多、应用日益复杂的各种网络相继出现, 使得 QoS 技术成为一个相当复杂并涉及诸多因素的技术, 实现 QoS 控制的难度大大增加。尽管各种通信网络上各种应用的 QoS 往往具有不同的结构和特点, 其中相当大一部分问题却可以简单归结为在端到端网络上如何更有效传输数据的问题。换句话说, 就是在具有带宽、缓存、噪声等约束条件下的实际网络中, 如何在保证基本质量的同时, 提高一部分重要数据的传输质量。针对这一类问题, 提出并实现了一种切实可行的适用于具有上述特征的端到端高速网络的 QoS 解决方案。

1　参数模型

QoS 技术是一种控制技术, 目的是结合网络上的实际状况控制网络参数, 使其更符合应用要求。正确给出合适的网络参数, 有利于 QoS 技术的高效实现。在各类网络中, 对各种不同的应用需求, 需要分析的网络参数往往略有不同, 但在通常的端到端网络中, 要实现 QoS 控制, 下列参数都是十分重要的[2,3]。

(1) 有效带宽 B_e 有两个含义, 一个是指面向连接传输协议(如 TCP)中一对对等通信实体中较小的那一个带宽值; 另一个含义是指进行一项通信服务时实际传输数据流量对传输时间的变化率。若在同一时刻内, 网络上运行有多个服务, 服务器端的有效带宽需要满足下列约束条件:

$$B \geq \sum B_e \tag{1}$$

其中 B 是服务器的可用带宽。

(2) 闲置带宽 B_i 是指网络一端的可用带宽与当前所有服务中正在使用的有效带宽的差额, 即

$$B_i = B - \sum B_e \tag{2}$$

(3) 服务带宽 A_e 是指网络中某个具体服务在单位时间内的数据流量。

(4) 端到端网络延时 τ是指单位数据由一端传输到另一端所花费的时间与预计时间或者服务要求时间的正差值。

(5) 应答时间 γ是指网络通信中一端从发送请求到收到第一份应答之间的时间差。

(6) 抖动 q 是指网络延时随时间的变化率。

(7) 数据包容量 P 是指包交换网络中每个数据包的字节数。

(8) 数据包丢失率 σ是指在网络传输中数据包遗失或者出错的比率。

(9) 无效带宽 B_u 是指相对于有效带宽的数据包丢失和错误所占用的那一部分带宽。

2　控制方法

2.1　QoS 控制的数学模型

任何一种 QoS 技术的实质都是根据已知的和(或)可操作

收稿日期: 2005-03-17; 修返日期: 2005-04-26

的参数 K_r 来计算未知的和(或)不可操作的参数 U_r,并根据计算结果改变 K_r 从而影响和控制 U_r,使得通信网络的服务质量得到改善。设网络质量用增量 S 来衡量,则

$$S = Q\{K_r \cup U_r\} \quad (3)$$

已知在一个典型的网络中,状态参数通常包括有效带宽 B_e、服务带宽 A_e、网络延时 τ 抖动 q 以及丢包率 σ 等。由于这些参数一般是时变的,所以还应引入一个时间参数 t,于是可得 QoS 原型方程(组):

$$S = Q\{B_e, A_e, \tau, q, \sigma, t\} \quad (4)$$

如果 S 本身对于 t 而言是平稳的,则可记

$$S(t) = Q(B_e, A_e, \tau, q, \sigma\} \quad (5)$$

在通常情况下 $B_e \in K_r$, A_e, τ q, $\sigma \in U_r$。显然,在一个实际网络中,一定存在一个极大值 S_{max},使得不论 B_e, A_e, τ q, σ 如何取值,总有 $S(t) \leq S_{max}$。如果 S 是连续的,则当 $S = S_{max}$ 时,必然有如下等式成立:

$$\frac{\partial S}{\partial B_e} = 0 \quad \frac{\partial S}{\partial \tau} = 0 \quad \frac{\partial S}{\partial q} = 0$$
$$\frac{\partial S}{\partial \sigma} = 0 \quad \frac{\partial S}{\partial A_e} = 0 \quad (6)$$

称式(6)为 QoS 状态方程组。通过它可以看出,如果 QoS 模型是一个多项式模型,并且参数的最高次数不大于两次,则网络存在一个可求的最优状态。求解这个状态实际上只需求出由式(6)构成的线性方程组的解即可。然而目前并非所有 QoS 问题都可以简化为上述线性解问题。因此,解决更为复杂的 QoS 问题需要寻找其他一些简单可行的方法。

(1) 迭代法

用迭代法解 QoS 状态方程组是一种有效的方法,而且它适用于 QoS 控制过程。上面的分析表明,QoS 控制系统本质上是一种反馈系统,它先根据可直接操作的参数 K_r 去求解不可直接操作的参数 U_r,然后根据计算结果反馈并调整 K_r,再根据调整后的 K_r 去求解新的 U_r,反复计算直至得到满足条件的解。这实际上是一种迭代过程,如果迭代是收敛的,那么网络状态就是稳定的,应用该模型的 QoS 控制也是可实现的;否则表明网络状态异常,或者相应的 QoS 模型不可实现。

(2) 局部搜索法

直接利用迭代法求解状态方程组的难点是需要对 QoS 状态模型进行调整,并且还要选择合适的迅速收敛的迭代算法。而局部搜索法则更为直接地采用了“迭代改进”的原理,它在 QoS 原型方程(组)的解空间中一次搜索一个点,每次迭代都从当前点的邻域中选取一个新点,如果这个新点的评估值优于当前点,则用它取代当前点,否则就选取当前点邻域中的其他点来进行比较。该方法在没有进一步改进的可能或者满足终止条件的时候终止。

(3) 超越局部搜索法

局部搜索法通常能够简单有效地处理许多 QoS 问题,然而局部搜索法的不足也很明显,它只能提供局部最优解,这些最优解还依赖于起始点的选取。在很多 QoS 问题中,仅仅得到局部最优解还不能满足要求。遇到这种问题时,就不得不从大量不同的起始点来开始局部搜索,并寄望于在这些点中总有一些会通向最优解或者足够令人满意的解。然而,在局部搜索法中缺乏一种通用的过程来界定它的解与全局最优解之间的差距有多远,所以当总是得不到令人满意的解时,依然不能推断这个解不存在,除非遍历了 QoS 问题的异常庞大的解空间。因此,在局部搜索法失效的情况下,需要寻找更好的解决方案。模拟退火算法或者禁忌搜索算法提供了超越局部搜索法的较好方案。

2.2 控制服务带宽的算法

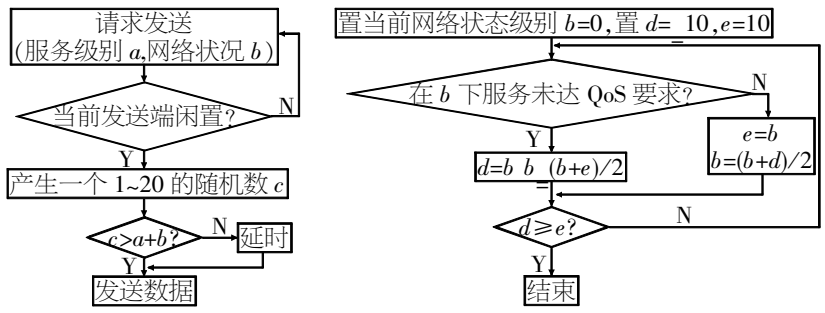
QoS 服务实际上是一种差别服务。为端到端网络提供 QoS 服务,其基本行为是为不同级别的服务分配不同的带宽。根据约束式(1)以及式(2),在一个基本上满负荷运行(闲置带宽 B_i 很小)的端到端网络中,要增加某些服务的带宽,就势必要减小另一些服务的带宽。这个问题首先归结为如何按照意愿来控制指定服务带宽。因此,必须根据问题的复杂度选择最合适的算法,迭代法和局部搜索法是解决此问题较好的方法。

注意到如果能够找到有效的方法控制服务带宽 A_e,令高级别的用户更容易得到高的服务带宽,从而提高他们的数据传输率,那么就实现了 QoS 控制的基本要求。

本文实验发现,通过改变包交换网络中的数据包大小,可以显著改变服务带宽:数据包大小在 10KB 左右的时候,服务带宽约为 40kbps;数据包大小在 3KB 左右的时候,服务带宽约为 16kbps。事实上,通过理论分析也能得出同样的结论。总之,可以通过控制数据包容量的方法来简单地控制有效带宽 B_e,限制某些服务的带宽,从而将可用带宽转移到其他高级别的服务上去。

除了改变数据包容量之外,另一种可有效改变服务带宽的方法是直接在发送端阻塞发送线程。在发送端某一时刻同时运行的多个服务,要发送数据,通常需要竞争发送端的使用权。基本的竞争调度算法一般在网络层或者传输层上实现,根据协议的不同,竞争机制也会有所不同。但现有的这一类协议并没有在争夺端口使用权时考虑 QoS 控制的需求。在应用层上实现的第一种算法 A1 是一种支持 QoS 的竞争调度算法。在这种算法中,将 QoS 服务分为 21 个级别,每个级别在竞争发送权时处于不同的地位,其中 0 级是最高级别。一般情况下,0 级用户在需要发送数据时,总是能够获得空闲的发送端;其他级别的各个用户,按照级别高低,获得空闲发送端的概率不同。该算法的具体过程为:用整数 0 ~20 来表示 QoS 级别,用整数 - 10 ~10来表示网络状况等级。当一个具有级别 a 的服务在状况为 b 的网络上请求发送数据时,如果当前发送端空闲,则先掷一颗 20 面的骰子,用掷出的点数 c 减去 a, b 之和,如果结果不小于零,则马上发送数据;否则将发送线程挂起一段时间(具体时间长短可根据情况设定)。程序流程图如图 1(a)所示。其中表示网络状况等级的全局参数(- 10 ~10: - 10 表示最高级,0 表示正常,10 表示最低级)可根据当前网络实际情况进行调整:如当前网络状况好,则调整该参数到一个较小数值,可保证运行中的大多数级别的服务都享有较大的服务带宽;反之,则将其调整到一个较大数值,减小级别较低服务的带宽,以保证级别较高服务仍可获得较高带宽。该算法通过概率方法产生延时来控制服务带宽,实现较简单。通过实验可证明该算法的有效性:服务级别为 10,服务带宽约为 42kbps;服务级别为 0(最高),服务带宽为 155kbps;服务级别为 20(最低),服务带宽为 30kbps。算法 A1 可解决 QoS 服务的基本问题,即

通过为高级别用户提供较高的服务带宽来保证其服务质量。然而这个算法还可改进。它会带来一些额外开销,因为这个算法在每次发送前总是会为低级别服务引入额外延时,从而使网络闲置带宽增大,网络总负载率降低。在网络状况较好时,可通过提高网络状况参数来适当减小这个开销。本文实现的第二种算法 A2 就是利用搜索方法在满足所有服务要求的条件下尽可能地减少额外开销,其流程图如图 1(b) 所示。算法 A2 仍有局限性,它仅根据网络实际状态选择较好的参数,仅从全局上减小闲置带宽。



(a)算法 A1 的流程图　　(b)算法 A2 的流程图

图 1　流程图

下面提出的算法 A3 基于如下原理: 当网络状况允许的情况下,可以适当暂时提高一些服务的 QoS 等级,以减小闲置带宽; 当网络状况不好的时候,则将提高的等级降回,以保证服务安全。算法 A3 的过程伪代码如图 2(a) 所示(其中, *b* 表示网络状态级别, s[ ] 表示当前运行的每一个服务, s. a 表示具体服务的等级, s. id 表示具体服务的标志, *i* 表示闲置带宽) 。事实上,算法 A3 是典型的最陡上升爬山法,是一种有效的局部搜索法,通过它可求出一个使得闲置带宽较小的局部最优解。然而,使用算法 A3 仍不能保证得到的解一定是全局最优的,也不能确知这个局部最优解和真正最优解之间相差多少。尽管如此,在绝大多数情况下,利用算法 A3 已能得到足够优解,完全可符合要求。下面提出的利用模拟退火算法的算法 A4 是一个比算法 A3 更好的选择。与算法 A3 相比,算法 A4 将得到更优的解。算法 A4 的过程伪代码如图 2(b) 所示(其中的符号意义同算法 A3) 。

```
procedure
begin
  get( b) //用算法 A2 求 b
  get( i) //通过 s[ ] 和 b 求 i
  while ( b < s. a) ( for some s in
s[ ]) /* 判断是否达到每个服
务的要求* /
    min = i
    for each s in s[ ] ( b < s. a)
      s. a = s. a - 1
      get( i)
      if i < min
        min = i
        index = s. id
      end if
      s. a = s. a + 1
    end for
    s[ index] . a = s[ index] . a - 1
  end while
end
```

(a)　算法 A3 的伪代码

```
procedure
begin
  init T //初始化温度
  init t//初始化退火时间
  get( b) //用算法 A2 求 b
  get( i) //通过 s[ ] 和 b 求 i
  while ( b < s. a) ( for some s in s[ ])
  //判断是否达到每个服务的要求
    min = i
    while T > 0
      for each s in s[ ] ( b < s. a)
        s. a = s. a - 1
        get( i)
        if i < min
          min = i
          index = s. id
        else if random[ 0,1] < e^((i - min)/T)
          min = i
          index = s. id
        end if
        s. a = s. a + 1
      end for
      T = g( T, t)
      t = t + 1
      s[ index] . a = s[ index] . a - 1
    end while
  end while
end
```

(b)　算法 A4 的伪代码

图 2　伪代码

3　解决方案的实现

根据上述研究结果,本文设计实现了一个有效的、可扩展的 QoS 控制核心(类库) ,并且在这个核心上实现了一个应用实例。

3.1　状态参数的提取

在端到端网络中应用 QoS 技术,第一步要做的就是获得网络状态参数。在这里选取的核心参数是有效带宽 B_e 以及服务带宽 A_e。注意到这两个参数都与时间有关,需要通过实际测试来确定它们。

要获得当前的有效带宽 B_e,可以建立连接,发送测试数据包,再根据反馈结果计算有效带宽。其基本原理是在接收和发送时均预先打上时间戳,然后根据时间差值计算出传输时间 Δt,再根据发送的字节数和传输时间计算出有效带宽 B_e。本文编写代码定义了用于测试网络状态参数的测试数据包类,它继承自 3.2 节中的网络数据包类; 同时本文编写代码描述了 QoS 参数接口,定义了闲置带宽、有效带宽、平均数据包容量、平均响应时间以及延时等参数。要建立合理的网络状态参数,就需要对获得的测试数据进行分析。前面提到过,可用收发一个数据包的大小和时间间隔的比值(变化率) 来表示有效带宽。假设收到一个测试数据包,长度为 1kb,所花时间为 1ms,那么可认为当前连接的有效带宽为 1kb/1ms = 1Mbps。这种测试有效带宽的方法比较简单,通过这种方法可迅速得到给定时刻的网络状况。这种通过计算有效带宽来判断网络状态的方法有一个缺点,即用它只能测量任意给定时刻的网络状况,而不能对下一时刻的网络状况进行预测。一种改进方案就是利用线性预测技术,用当前时刻之前的一段 Δt 时间内所测得数据的加权平均值去预测下一时刻的有效带宽; 另一种改进方法则是用平均有效带宽去代替瞬时有效带宽。这两种方法各有其特点。一般来说,在网络状况比较稳定,服务带宽基本不随时间变化的时候,用第二种方法比第一种方法预测得更准确; 如果有效带宽随时间有较大变化,则用第一种方法会比较好。

获得服务的有效带宽 B_e 之后,很容易计算得到实际的服务带宽 A_e。前面提到过,服务带宽就是某具体服务在一段时间内接收数据量和时间的比值,它通常和有效带宽的调和平均值正相关。通过有效带宽求平均值来表示服务带宽或者根据定义来求服务带宽都是可行的。

3.2　QoS 控制的实现

在获得了准确的有效带宽和服务带宽数据之后,服务器就可以处理 QoS 请求,并按照所承诺的级别来收发数据了。端到端网络通信需要一定的传输协议。设计并编程实现的 QoS 控制核心在理论上可以用任意传输服务进行封装。在实验中采用 TCP 协议进行封装。①为包交换网络设计基本的数据包结构,定义网络数据包类,其中包括标志 id、名称 name、段号 segment_number、容量 segment_capacity、总段数 num_of_segment、总大小 total_size 以及实际数据 data; ②封装 QoS 请求信息,构造 QoSRequest(QoS 控制请求) 类,由客户端向服务器端发送请求服务等级 requestLevel、延时约束 latencyConstraint、已传输时间 transportTicks、已收到的数据包 packageReceived、剩

余数据包 packageLeft 等数据; ③封装 QoS 应答数据, 构造 QoSResponse(QoS 应答信息) 类, 由服务器端告知客户端所提供的服务级别 serviceLevel 以及提供该级别服务的对象标志 providerID; ④用套接字封装 TCP 协议用来发送各种数据, 这部分程序是本解决方案的关键。在实际应用中, 客户端与服务器建立连接的同时发送 QoS 请求, 服务器端响应请求, 设定 QoS 级别, 在之后的数据传输中, 按照该级别的规则或约定来传送数据。这样就实现了一个 QoS 的基本构架。这个完整的 QoS 核心包括网络数据、协议、套接字和异常处理等各个部分。

3.3 应用实例: 网络图片发布/浏览程序

这里利用上面构建的 QoS 核心实现了一个应用实例: 网络图片发布/浏览程序。该程序可让用户通过网络欣赏到图片, 并根据用户请求的或服务器允许的服务级别来获得相应质量的服务。服务器一方在服务器上启动服务端浏览器 ImageServer 发布一张图片, 然后用户通过客户端 ImageClient 连接到服务器, 可载入图片欣赏。而在载入图片之前, 该用户可设定服务级别和传输延时约束, 如果服务级别被接纳, 则按该级别提供服务; 否则服务被拒绝。在提供服务时, 根据请求的级别和延时约束, 按照约定提供尽可能好的服务。服务结束后, 在客户端浏览器 ImageClient 中还可返回网络状况记录等信息 (图 3)。

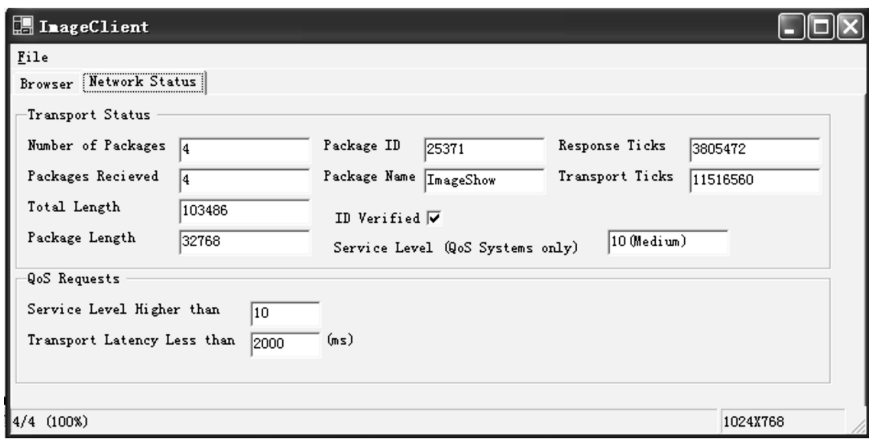
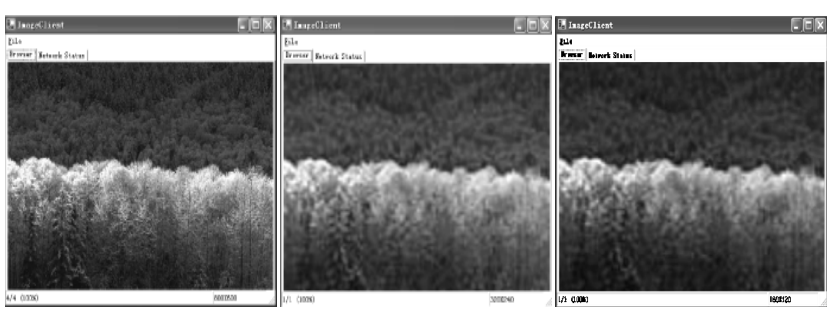


图 3 客户端浏览器设置和信息界面

图 4 是本应用的一个例子, 其中图 4(a) 的请求为服务级别 0(最高) , 延时约束 2s, 应答为响应时间 140ms、传输时间 80ms、服务质量 800 ×600 像素; 图 4(b) 的请求为服务级别 10 (一般) , 延时约束 0. 6s, 应答为响应时间 172ms、传输时间 323ms、服务质量 320 ×200 像素; 图 4(c) 的请求为服务级别 20 (最低) , 延时约束 0. 8s, 应答为响应时间溢出、传输时间忽略、服务质量 160 ×120 像素。从图 4(a) ~图 4(c) 的对比可见, 用 QoS 核心实现的图片发布/浏览程序可以方便地根据客户的实际情况调整服务质量, 在满足客户要求的情况下尽可能地提供较好的服务, 很好地实现了 QoS 服务的要求。由此充分验证了本解决方案算法的正确性与可行性。



(a) (b) (c)

图 4 一个网络图片发布/浏览应用的例子

3.4 扩展

本文在设计 QoS 系统时尽量将控制核心独立出来, 一个重要原因是希望该系统尽可能地具备可扩展性。从过去到未来, 从局域网到国际互联网, 从有线网到无线网, 从电信网络到移动通信网络, 从文件传输协议(FTP) 到流媒体实时传输协议(RTSP) , QoS 的需求在不断变化。一个设计合理的 QoS 系统应该是独立于设备、环境、协议和服务的。在面临任何新条件、新需求时, 系统总能方便地扩展新的功能, 并且在形式上与原系统保持统一的接口和兼容性。本文将端到端系统独立和抽象出来, 将 QoS 请求和应答独立出来, 都是为了实现松散耦合, 实现不依赖于具体环境和协议的 QoS 控制系统。尽管目前本文所设计的系统是运行于 TCP 协议之上的, 但它同样可以用在 UDP, ICMP, IGMP 等各种协议上, 并保证接口行为的一致性。

4 结论

本文分析讨论了端到端 QoS 控制的原理, 提出了通过控制服务带宽实现端到端 QoS 的方法。利用控制数据包容量和服务器端阻塞发送线程的方式限制低级别服务的带宽, 从而保证高级别服务拥有更高的带宽。根据概率原理实际设计出简单易行的竞争调度算法, 并探索了进一步优化的可能, 提出了用于提高带宽利用率的局部搜索和全局优化的手段。在此基础上, 本文编码实现了一个 QoS 控制核心, 保证了它的有效性, 并使之在将来有利于扩展。在核心之上, 还设计实现了一个应用实例, 进一步展示和验证了设计成果。

QoS 涉及的领域是非常广阔和复杂的, 并且 QoS 的应用将会随着网络技术的发展不断深入。在不远的将来, QoS 技术将会成为整个网络的基础技术, QoS 服务将取代 "尽力而为 "的服务成为网络的基本特性。从全局和抽象的高度把握 QoS 技术的核心对于理解和应用这种技术具有重大意义, 通过研究端到端网络的 QoS 解决方案把 QoS 技术推到一个系统的层次上也正是研究的意义所在。

参考文献:

[1] 日] 户田岩. Quality of Service(网络服务质量技术) [M]. 朱青. 北京: 科学出版社, 2003. 2-6.

[2] Andrew Campbell, Geoff Coulson. A QoS Adaptive Transport System: Design, Implementation and Experience[C] . Boston: Proceedings of the 4th ACM International Conference on Multimedia, New York: ACM Press, 1997. 117-127.

[3] Martin Reisslein, Keith W Ross, Srinvas Rajagopal. A Framework for Guaranteeing Statistical QoS[J] . IEEE/ACM Transactions on Networking, 2002, 10(1) : 27-42.

作者简介:

张昱(1974-) , 男, 山西沁水人, 讲师, 博士, 研究方向为网络信息处理及 Internet 技术、等离子体生化作用系统电子学等, 已发表论文十余篇。