队列调度算法在网络中的应用研究

杨永斌　唐亮贵
（重庆工商大学计算机科学与信息工程学院　重庆 400067）

摘　要　作为保证QoS的一种重要手段，队列调度算法近年来引起了网络研究者的广泛关注。本文首先介绍了队列调度问题及一些常用的队列调度算法，然后提出一个非GPS模型的队列模型及调度算法——WDQ算法(Weighted Delay Queuing，基于权重的延迟队列)，并且解释了这种算法能够有效抵抗通信量的突发，具有控制不同权重分组延迟的能力，对于提高和改善网络服务质量QoS方面的研究和网络运行情况的研究具有积极意义。

关键词　队列，调度，QoS模型，WDQ

The Application Research of Queue Scheduler Algorithm in Network

YANG Yong-Bin　TANG Liang-Gui
(College of Computer Science and Information Engineer, Chongqing Technology and Business University, Chongqing 400067)

Abstract　As an important means to ensure QoS, network researcher has paid more attention to queue scheduler algorithm recently. At first, this article introduces queue scheduler question and a few queue scheduler algorithm in common use, then presents a non-GPS queue model and scheduler algorithm-WDQ algorithm(Weighted Delay Queuing), and explains that this algorithm can resist the outburst of communication, it also has an ability to control the delay of packets with different weights, it has more meaning for the research of advancing and improving network QoS and network running status.

Keywords　Queue, Scheduler, QoS model, WDQ

1　引言

随着网络应用的日益广泛，一个通信网络往往需要支持多种业务，每种业务意味着不同的服务质量(QoS)要求，QoS是指网络在传输数据流时要求满足的一系列服务请求及实现这些请求的机制，这些服务请求可用一系列指标来衡量：如带宽要求、传输延迟、延迟抖动、可靠性、丢失率、吞吐量等，网络的QoS能力与网络中采用的队列调度机制有着密切的关系，队列调度算法正是提供服务质量保证的重要机制之一。

一个网络可以视为一个非常复杂的排队系统，设计队列调度算法时要在延迟、实现复杂性和公平性之间进行综合考虑，以实现服务质量保证并优化网络资源利用率。

2　问题的提出

对于一个路由器或交换机，接入端口有 n 个队列，每个队列有一个固定的权重，用一个调度器为这 n 个队列服务。如何使得进入各个队列的分组在队列中的延迟是可控的，同时保证权重越大的分组的延迟越小？并且考虑到计算机的通信量常常是有突发的情况，希望在这种情况下，分组的延迟不会有太大的抖动。近年来，基于GPS模型的公平队列的调度算法得到了广泛的研究，以WFQ(Weighted Fair Queuing，基于权重的公平队列)最为经典，然而由于通信量的突发，使得分组的延迟存在着较大的抖动。为解决此问题，本文提出一个非GPS模型的队列模型及队列调度算法——WDQ算法(Weighted Delay Queuing，基于权重的延迟队列)。

3　队列调度问题及常见调度算法

随着网络应用的迅速发展，其规模越来越庞大，结构日趋复杂，仅仅依靠端到端的拥塞控制是不够的，网络本身也必须参与资源的控制和管理，在网络发生拥塞时，网络节点必须丢弃一些分组，这个问题的解决首先必须实施有效的队列管理机制。传统排队论的分析侧重于不同到达过程和服务时间的业务流的排队时延特性，根据对实际系统的抽象，分析时常用到的服务策略有FIFO、优先级机制等。在实际应用中，网络状况非常复杂，队列的服务策略问题就被单独地提出来，需进行专门的分析，它是进行排队分析的基础，对它的研究是优化排队性能的重要手段。队列调度算法就属于服务策略研究的范畴。在应用中，队列调度算法运行在网络节点中发生冲突需排队等待调度之处(如图1所示)，它的任务是按照一定的服务规则对交换节点的不同输入业务流分别进行调度和服务，从这多个队列中选择下一个要传输的分组，使所有的输入业务流能按预定的方式共享交换节点的输出链路带宽，并满足不同业务的不同服务质量(QoS)要求。常见的队列调度算法主要有以下几种。

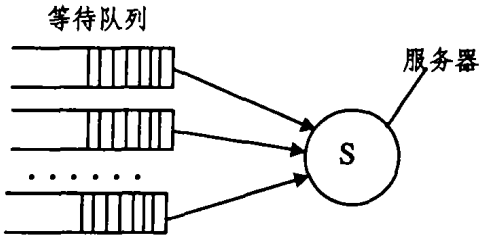

图1　队列调度的应用情形

3.1　先到先服务(First Come First Served)

用FIFO队列实现FCFS是网络应用中使用最多的一种方式，它的最大优点在于实施简单。FIFO本质上是一种“去

杨永斌　讲师，主要研究方向：计算机算法、计算机网络、数据库技术、网络教育。唐亮贵　讲师，博士研究生，主要研究领域为：智能多主体系统与分布式计算，智能电子商务理论与技术，网络计算与应用。

尾"(Drop-tail)算法,不需要选择丢弃的分组,在系统中没有空闲缓冲资源时就丢弃到达的分组。虽然这种算法已经在网络应用上成功工作了许多年,但它有三个严重的缺陷:①持续的满队列状态;②业务流对缓存的死锁;③业务流的全局同步。

3.2 随机早期检测算法(Random Early Detection)

RED配置在路由器上监视网络流量以避免拥塞,当即将发生拥塞时,它随机丢弃进来的分组,而不是等到队列缓冲区满时才开始丢弃所有进来的分组,这样可以最少化全局同步的发生。当拥塞发生时,RED丢弃每个连接分组的概率与该连接占用的带宽成比例,它监视每个输出队列的平均队列长度,随机选择分组丢弃。虽然RED通过随机早期检测和丢包,有效地在TCP流之间分配带宽,但当TCP数据流和非TCP数据流混合时,RED不能有效地保护TCP流,没有拥塞控制或采用比TCP更贪婪的非TCP流将比TCP流攫取更多的网络带宽,这种不公平性的主要原因是拥塞发生时非TCP流不降低发送速率或降低的程度比TCP少,而RED对所有的流都采用同样的丢包比率。

比起Drop-Tail,RED算法的两个好处是:首先,队列缓冲总是预留了一定的缓冲空间,这样可以更好地处理突发性。其次,保持较短的队列长度,可以更好地支持实时应用。

3.3 分组公平队列(Packet Fair Queuing)

PFQ算法是基于GPS(Generalized Processor Sharing)的算法。从20世纪80年代末以来,国际上对PFQ进行了大量的研究。PFQ能够保证连接的预约带宽、最大端到端时延以及时延抖动,是实现QoS的关键技术。目前已有WFQ(Weighted Fair Queuing)、WF2Q(Worst-case Fair Weighted Fair Queuing)、WF2Q+、SPFQ(Start-Potential Fair Queuing)、SCFQ(Self-Clocked Fair Queuing)等多种PFQ方案。

3.4 基于轮循的调度算法(Round Robin)

传统RR算法对不同队列(业务流)进行无区别的循环调度服务。这样,如果不同队列具有不同的分组长度,则分组长度大的队列可能会比分组长度小的队列接受更多的服务,使队列之间产生不公平的现象;而且这种算法不能为业务提供时延保证。为了改进RR算法的时延特性和在变长分组环境下的不公平性,出现了一些改进型的算法,如加权轮循WRR(Weighted Round Robin)、差额轮循DRR(Deficit Round Robin)、紧急轮循URR(Urgency-based Round Robin)。这些算法力求在尽量保持RR算法实现简单性的同时,从不同的方面改进RR算法的时延特性及其在可变长分组环境下的不公平性。

4 队列模型

QoS的目的是为不同等级的业务流提供不同的服务质量。从应用程序看来,服务质量最重要的是带宽和延迟,具体表现为对于路由/交换接口,按照权重分配转发能力。处理机共享(GPS)模型是一个理想化的流模型,它根据各队列的共享比例对所有的活动队列同时服务,不存在非抢占(no preemtion)的单元。在任何时间间隔内,如有M个非空队列,服务器就按照一定的服务比例对这M个队列同时进行服务,GPS对每个队列业务流保证有明确的端到端的时延上限,而与其他队列业务流无关。但在实际的分组系统中:在任何给定的时刻只能有一个分组可以得到服务,并且在某一时段内服务是不能够被打断的,因此出现了基于分组的GPS模型(P-GPS)。

对于实时程序中,分组的延迟比吞吐率更为敏感。所谓延迟,在一个接入路由器看来,就是一个分组进入该路由器的时刻和离开该路由器的时刻之差。考查GPS模型的WFQ、WF2Q算法,发现在过载(过载是指各队列分组到达的速率之和大于信道的带宽。如果没有过载,分组总能及时的转发,分析其延迟就没有意义)的时候,分组的延迟会有较大的抖动,从而在分组延迟的统计图上形成一些向上的毛刺,并且有时高权重的业务流的分组会比低权重的业务流的分组有更大的延迟。高权重的业务流因为太大的突发而导致本业务流的等待队列太长,以至比同时进入的低权重的业务流的分组有更多的等待时间。为了使分组的延迟不出现大的抖动和向上的毛刺,本文提出一种新的模型——队列模型,其例子也是本文所要验证的WDQ算法。分组延迟来自于路由器上的排队时延,队列模型就是这样一种自然而然的模型(如图2所示),在每个分组从代表不同业务流等级的输出队列进入硬件队列之前,先要插入调度队列的合适位置(根据权重和其他条件)。提出这种模型是基于以下考虑:

队列模型是在分组一进入队列的时候就确定下其最晚的转发次序,在WDQ算法中,其最晚的转发次序是按照其权重比例赋予的,以此保证在最坏的情况下,分组的延迟与权重是成比例的(假设转发每个分组所用的时间都是一样的)。

队列模型具有一定的能力应付一定范围内的突发的流量。高权重的队列的分组和低权重的队列的分组之间往往会余有空间,作为突发缓冲区,缓解高权重的分组突然大量增加造成的恶化;并且每个分组到达的时候,它在调度队列中的位置表示了它的最晚转发次序,不会因为高权重队列的分组的挤占而无限推迟。而WFQ和WF2Q只是当分组在队首时才能确定其转发的次序。WDQ算法是基于此队列模型的。

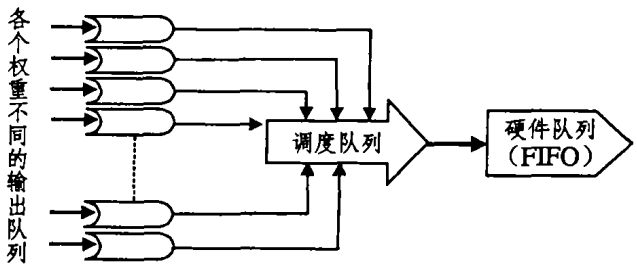

图2 队列模型体系结构

5 WDQ算法

5.1 算法介绍

WDQ是基于队列模型的分组调度算法,为了解释方便,先定义以下名词:

Ⅰ. Q_{last}:输出队列的最后一个分组的最晚调度次序号,每一个输出队列有一个,用来指向这个输出队列的分组在调度队列中的最后位置。

Ⅱ. Q_{max}:分组的最晚调度次序号,为此分组在调度队列中的次序号,每一个分组有一个,表示此分组的最晚调度次序。

Ⅲ. *SchedularQ*:调度队列,可以理解为一个链表结构。

Ⅳ. *MININDEX*:调度队列最小次序号,是一个全局变量。调度队列的所有分组的最小的索引号不能小于*MININDEX*,*MININDEX*是一个非负整数,一般初始化为0。

Ⅴ. $Q_{current}$：调度队列首位分组的次序号，是一个全局变量，用于更新 Q_{last}、Q_{max}。

Ⅵ. τ：调整的间隔时间，指的是调度队列 *SchedularQ* 的调整时间间隔。

由Ⅰ和Ⅱ知，$Q_{last} \geqslant \max\{Q_{max}\}$。

WDQ 算法具体描述如下：

假设有 n 个队列，其设定的权重比为 $2^{n-1}:2^{n-2}:\cdots\cdots:2:1$

1. 初始化：每个队列的 Q_{last}（按照权重从低到高）为 *MININDEX*，-1，-1，$-1\cdots\cdots$（-1 表示未定义，*MININDEX* 为非负整数，例如 0）。

2. 处理：每个分组到达其所属于的输出队列时，如果其队列还能接受该分组（队列的缓冲区还没有用完），那么在输出队列的尾部插入该分组，否则丢弃该分组。

3. 如果该分组没有被丢弃，根据该分组所在队列的上一个分组的最晚调度次序号 Q_{last}，计算出该分组的最晚调度次序号 Q_{max}：

a）如果 Q_{last} 不等于 -1，那么 $Q_{max}=Q_{last}+1$

b）否则这个队列的 Q_{max} 取为某个 Q_{last} 不等于 -1 的队列的 Q_{last}（例如权重最大的输出度队列的 Q_{last}）乘以二者权重比的积：$Q_{max}^{i-1}=Q_{last}^{j-1} * W^{j-1}/W^{i-1}$

然后调整 Q_{max}，在调度队列 *SchedularQ* 的适当位置插入该分组的地址或者拷贝，确保调度队列的各分组的 Q_{max} 是严格递增的，并更新相应输出队列的 Q_{last}。

4. 调度队列按照顺序转发分组，每次调度都是转发调度队列队首的分组。同时删去该分组在输出队列中的备份（如果在 3 中插入的是分组的拷贝）。

5. 每隔时间 τ，或者调度队列 *SchedularQ* 中分组的最大的 Q_{max} 大于输出队列的最大容量之和的若干倍（例如 10 倍，可以自己定义）的时候，则调整调度队列 *SchedularQ* 的所有的分组的实际最晚调度次序号 Q_{max} 和每个输出队列的最后一个分组的最晚调度次序号 Q_{last}：

a）$Q_{current}$ = 调度队列队首的分组的 Q_{max}；

b）调整输出队列的 Q_{last}：从权重最大的输出队列开始，权重最大的队列的 Q_{last} 减去 $Q_{current}$（$Q_{last}=Q_{last}-Q_{current}$），作为新的 Q_{last}；其余的队列的 Q_{last} 取原来的 Q_{last} 减去 $Q_{current}$ 的差，和权重最大的队列的新 Q_{last} 乘以二者的权重比（大权重比小权重）的大者作为新的 Q_{last}（$Q_{last}^{i}=\max\{Q_{last}^{i-\alpha/d}-Q_{current}, Q_{last}^{0} * (w_0/w_i)\}$，$i>0$，$w_0$ 是最大的权重）；

c）调度队列中的各分组的 Q_{max} 分别减去 $Q_{current}$，作为新的 Q_{max}，此时队首的 Q_{max} 为定义的最小索引号 MININDX。

WDQ 算法中，每个分组只要不被丢弃，当它进入输出队列时就可以确定其最晚转发次序，从而避免它的延迟无限增大。

5.2 性能分析

简单地说，队列模型的 WDQ 算法既有 PQ（Priory Queue）的特点，也有 FQ（Fair Queue）的特点。PQ 的特点表现在：每次对调度队列 SchedularQ 调整完，再次插入新分组的一段时间内（突发缓冲区未用完时），高优先级的分组总是排在低优先级的分组之前，即使这个高优先级的分组比低优先级的分组要晚到达。FQ 的特点表现在当突发缓冲区使用完之后，后来的高优先级分组依然会排到低优先级的分组后面，以避免低优先级分组的“饥饿”。

假设只有权重为 1 和权重为 2 的两个分组队列，并把所有的状态分为以下三类：状态 1、2、3（如图 3 所示）。其中状态 2 是有突发缓冲区的（阴影部分）；状态 1 无突发缓冲区，分组整齐的分为两部分，高优先级的分组排在前面；状态 3 无突发缓冲区，不同等级的分组交错排列。

按照 WDQ 算法的步骤 3 和步骤 5，由于不同业务流的分组总是交错到达，而不可能是一组组分别到达，因此状态 1 总会向状态 2 转变。同理，状态 3 总会向状态 2 转变。显然，如果高权重的业务流有突发流量，状态 2 会向状态 3 变化；如果没有突发流量，那么如果调整时间 τ 足够短，按照步骤 5，状态 2 依然是状态 2；否则，按照步骤 3，状态 2 向状态 1 变化。所以，如果高权重的业务流有突发流量，状态 2 会向状态 3 变化；否则，保持状态 2 不变或向状态 1 变化。

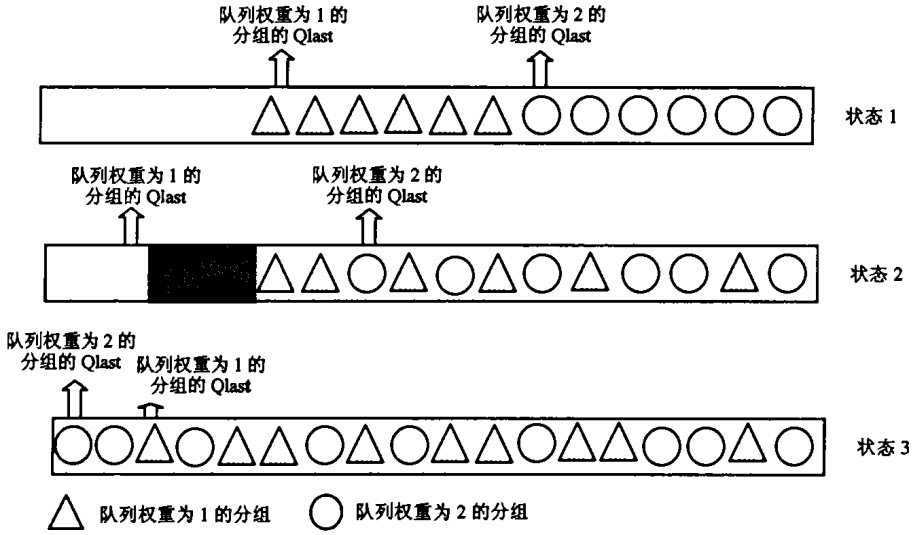

图 3 模拟仿真队列

状态 2 具有容忍突发的能力。因为在状态 2 中，权重为 2 的队列有了突发的分组，那么在权重为 2 的分组的 Q_{last} 和权重为 1 的分组的 Q_{last} 之间的空间为分组提供了缓冲（灰色部分），使得突发的高权重分组在一定程度之内不会比同时到达的低权重的分组更迟被转发。所以，使用队列模型的 WDQ 算法具有在一定程度内容忍通信量突发的能力。在状态 2 中，如果高优先级的业务流有突发的流量，突发的流量在一定范围内不会导致突发的分组比同时到达的低优先级的分组更迟被转发。WDQ 算法也具有一定的控制低权重的业务流的分组延迟的能力。每个分组在到达的时候，就确定其最晚转发次序 Q_{last}。这样在一定程度内，使得低权重队列的分组的转发时间

（下转第 124 页）

束算法描述。

算法1：COSC($task$, $authorized\_roles(task)$, $weight_{num}$, $weight_{sort}$, n)

1)初始化 $token=0, r_{cur}=NULL$;

2)如果 $token=n$,则算法成功结束,否则转3);

3)如果$\exists$ r,使得 $weight_{sort}(r, task)=token+1$,那么对$\forall$ $r'\in(r)_{RH}$,可以选择 r'激活任务 $task$,如果成功激活任务 $task$,则赋值 $r_{cur}=r, token=token+1$,转2),否则重复执行3);如果不存在这样的 r,则转4);

4)选择$\forall$ $r''\in(rcur)_{RH}$的角色激活任务 $task$,如果成功激活任务 $task$,则赋值 $token=token+1$,转2),否则重复执行4)。

下面以图3所示的公文处理工作流为例,使用算法1的步骤来解决批示办理任务中多个角色协同执行任务的序约束问题。为了阐述清晰,从图3中单独提出批示办理任务及其相应的授权角色和权值,如图5所示。

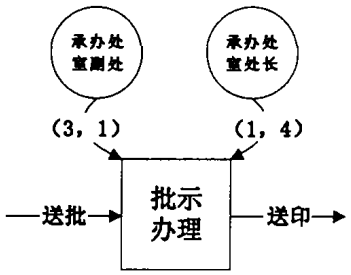

图5 应用算法1解决序约束问题的一个实例

1) 初始化 $token=0, r_{cur}=NULL$;

2) $n=4, token=0\neq n$;

3) $\exists$ $r=$承办处室副处长,使得 $weight_{sort}(r, task)=token+1=1$,假设其成功激活任务,则 $r_{cur}=$承办处室副处长,$token=token+1=1$;

2) $n=4, token=1\neq n$;

3) 不存在这样的 r,使得 $weight_{sort}(r, task)=token+1=2$;

4) 选择 $r''\in(rcur)_{RH}=$(承办处室副处长)$_{RH}$,假设 r''成功激活任务,则 $token=token+1=2$;

2) $n=4, token=2\neq n$;

3) 不存在这样的 r,使得 $weight_{sort}(r, task)=token+1=3$;

4) 选择 $r''\in(r_{cur})_{RH}=$(承办处室副处长)$_{RH}$,假设 r''成功激活任务,则 $token=token+1=3$;

2) $n=4, token=3\neq n$;

3) $\exists$ $r=$承办处室处长,使得 $weight_{sort}(r, task)=token+1=4$,假设其成功激活任务,则 $r_{cur}=$承办处室处长,$token=token+1=4$;

2) $n=4, token=4=n$,算法成功结束。

结束语 工作流与基于角色的访问控制均是目前计算机领域研究的热点问题。一个安全的访问控制模型是工作流管理系统中必不可少的重要组成部分,也是当前工作流研究的一个重要内容。由于传统的RBAC访问控制模型已经不能表达复杂的工作流安全访问控制约束,因此本文基于传统的RBAC模型,提出了一个新的基于双权角色的条件化RBAC访问控制模型CRDWR,阐述了基于动态角色分配的条件化RBAC策略,定义了基于双权角色的工作流系统访问授权新概念,并针对多个角色协同执行任务的序约束问题给出了基于令牌的序约束算法。该模型实现了角色的动态分配,实现了角色执行任务的事务完整性约束和多个角色协同执行任务的序约束,能够表达复杂的工作流安全访问控制约束。CRDWR模型还有许多需要研究和完善的地方,例如,访问授权的最小权限原则的执行算法,同一个角色多次激活任务的序约束,以及条件化RBAC访问控制模型中的角色冲突消解等问题,都有待于进一步深入研究和探讨。

参 考 文 献

1 Hollingsworth D. Workflow Management Coalition: The Workflow Reference Model [R]. Document Number WFMC-TC00-1003, Brussels, 1994

2 Ferraiolo D F, et al. Proposed NIST Standard for Role-Based Access Control [R]. ACM Transactions on Information and System Security, 2001,4(3):231～241

3 Wang X M, Zhao Z T, Hao K G. A Weighted Role and Periodic Time Access Control Model of Workflow System [J]. Journal of Software, 2003

4 Fan Y S. Foundation of Workflow Management Technology [M]. Beijing: Tsinghua University Press & Springer Press, 2001

5 Workflow Management Coalition: Workflow Management Coalition Terminology [R]. Document Number WFMC-TC-1011, Brussels, 1996

6 X. 812 ITU. Security Frameworks for Open Systems: Access Control Framework [R],1995

(上接第58页)

不会被高权重队列的分组任意挤占而无限延长,有利于减少分组延迟的向上的毛刺。利用NS2仿真模拟实验能有效验证和解释上述结论。

结束语 网络对QoS的保障需要一套完善的框架和机制,队列调度算法在各个层次上有力地支持了QoS保障的实现。队列调度算法可以在业务流聚合过程中发挥重要的作用:对于边缘路由器,业务状况复杂,通常会根据目的地或业务QoS等级进行业务流聚合,同目的地或同QoS的分组进入同一个队列接受同等级的服务,队列调度算法可以解决各个队列的分组如何汇聚成一条注入流的问题。

队列调度在QoS保障方面发挥着重要作用,同时在拥塞控制、高速路由、交换等方面也是必须考虑的问题。目前,基于GPS的调度算法仍是研究的热点之一,同时出现了基于服务曲线的算法、基于模糊逻辑的队列管理算法等多种新的调度思想。随着网络复杂性的增加和人们对网络理解的深化,队列调度的研究将更加深入,对应用的指导也将更加广泛。

参 考 文 献

1 林闯,等著.计算机网络的服务质量(QoS)[M].北京:清华大学出版社,2004

2 (美)Flannagan M,等著.尹敏,等译.Cisco Catalyst QOS—园区网中的服务质量[M].北京:人民邮电出版社,2004

3 陆慧梅,向勇,史美林. Internet QoS研究[J].小型微型计算机系统,2002,23(7):786～791

4 王重钢,隆克平,龚向阳,程时端.分组交换网络中队列调度算法的研究及其展望[J].电子学报,2001,29(4):553～59

5 姜宁康,李毓麟.NS网络仿真技术及其应用分析[J].小型微型计算机系统,2002,22(4) :415～417

6 金烨,樊骅.NS中数据流传输处理机制分析及流分析器的实现[J].计算机工程与应用,2003(22):153～155