图书馆多媒体业务流的短时公平分组反馈调度方法①

宋路露

(广东工业大学 图书馆, 广州 510006)

摘 要：研究了一种新的基于短时公平的分组调度算法的问题. 基于短时公平性的分组反馈调度算法改进了WF2Q+算法在短期内无法为新加入客户端提供公平服务的缺陷, 增强了调度算法的适应性和公平性. 在本算法中, 调度器中的各个客户端权值能够根据其获得的实际服务量状况在线调整, 增强了系统的鲁棒性和自适应性, 同时提高了系统实现短期公平性的能力, 对各个客户端提供更为公平的服务质量（Quality of Service, QoS）.

关键词：反馈的调度, 短期公平, WF2Q+, 分组调度

Feedback-Control Based on Short Term Fairness Scheduling Algorithm for Digital Libary

SONG Lu-Lu

(Library, Guangdong University of Technology, Hefei 510006, China)

Abstract: In this paper, we have proposed a new packet scheduling algorithm for short term fairness in this article. A feed-back control mechanism is proposed to improve the network packet scheduling algorithm. The weights of each client in the scheduler can be adjusted and controlled online according to the actual amount of service with the feedback mechanism. As a result, the robustness and the adaptability of network scheduling system have been enhanced. The capability of short term fairness has been improved, and each client can obtain a better fair QoS (quality of service) in network systems.

Keywords: feedback-based scheduling; short term fairness; WF2Q+; packet scheduling

随着数字化图书馆业务流的日渐扩大, 各个客户端的数据流在网络中传输的行为, 主要是受到各个路由器等的中间节点的影响. 路由器在网络当中不但担当快速转发的功能, 而且还起到连接广域网关键路径的网关作用. 同时, 路由器的功能主要由软件系统来实现, 便于修改, 因此在路由器上实现 QoS 保证机制更具现实性. 但是, 路由器只能够对到达的数据包一个一个地转发, 所以目前对路由器的 QoS 保证机制的研究主要集中在分组调度算法的研究和队列管理（缓存管理）两个大方面上.

早期的调度算法只有先进先出（First In First Out, FIFO）排队算法实现的先来先服务. 但是这种排队算法既不利于带宽的合理利用, 也无法体现各个客户端数据流的公平性, 只是一种非常简单的排队方法. 后来, 产生了优先级排队（Priority Queuing, PQ）算法和通用处理器共享（Generalized Processor Sharing, GPS）的公平算法等一些分组调度算法. 其后, 在 GPS 算法的公平分组调度机制的基础上, 众多学者提出了一系列的分组公平队列(packer fair queuing, PFQ)[1][2][3]算法, 例如使用最小虚拟完成时间机制（Small virtual Finish time First, SFF）的加权公平队列算法（Weighted Fair Queuing, WFQ）[4][5], 实用最小虚拟开始时间机制（Small virtual Start time First, SSF）[6][7]的随即公平队列算法（Stochastic Fair Queuing, SFQ）[8], 使用最小合格虚拟完成时间机制（Small Eligible virtual Finish time First, SEFF）WF2Q+[12]等.

WF2Q+调度算法主要集中在保证系统的长期公平性和系统的最大吞吐量. 在这个方面, WF2Q+的性

① 收稿时间:2013-10-17;收到修改稿时间:2014-03-17

能已经得到了不少实践的验证. 但是, WF2Q+算法在实现短期公平性上还存在着一个严重的问题.

为了要实现对新加入客户端的短期公平服务, 增强系统的鲁棒性和自适应性, 我们基于WF2Q+内核发明了一种客户端权值在线自动调整的自适应分组调度策略—基于短时公平性的分组反馈调度方法.

1 系统概述

1.1 服务管理模型

对于网络传输中的调度机制, 可以将其看成是一种对客户端业务请求的管理. 图 1 中给出的模型.

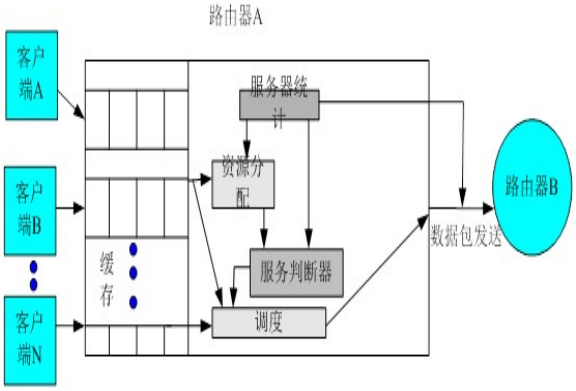

图 1 服务管理模型图

1.2 系统的体系结构

根据业务管理模型,我们设计了一个涉及数据包管理的反馈控制系统. 系统的控制框图如图 2.

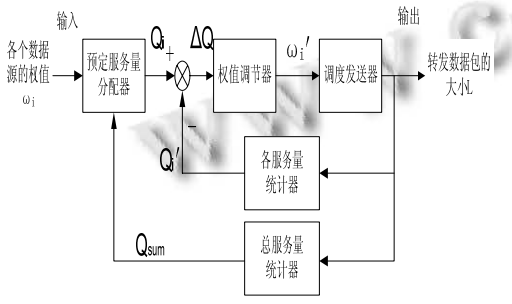

图 2 分组反馈控制系统框图

图 2 中, 整个反馈控制系统以各个数据源的权值作为输入, 通定服务量分配器, 根据当前系统总服务量. , 估算各个数据源当前的服务量期望值; 然后, 权值调节器通过期望值和各数据源实际服务量的比较, 对各个数据源的权值进行调整, 生成新的调度权值供调度发送器作为调度依据; 最后, 调度发送器根据最小合格虚拟完成时间优先的机制(SEFF 机制), 按照各个数据源的权值进行选择, 安排数据包发送的先后顺序.

1.3 WF2Q+算法的详细说明

基于SEFF 调度机制WF2Q +算法可以实现比GPS更公平和更大的输出. 控制模块显示在图 3.

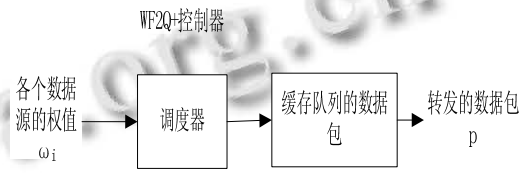

图 3 WF2Q+控制框图

它主要包括三个过程: 初始化, 进入队列和离开队列. 这个流程图的数据包进入队列和离开队列分别显示在图 4 和图 5.

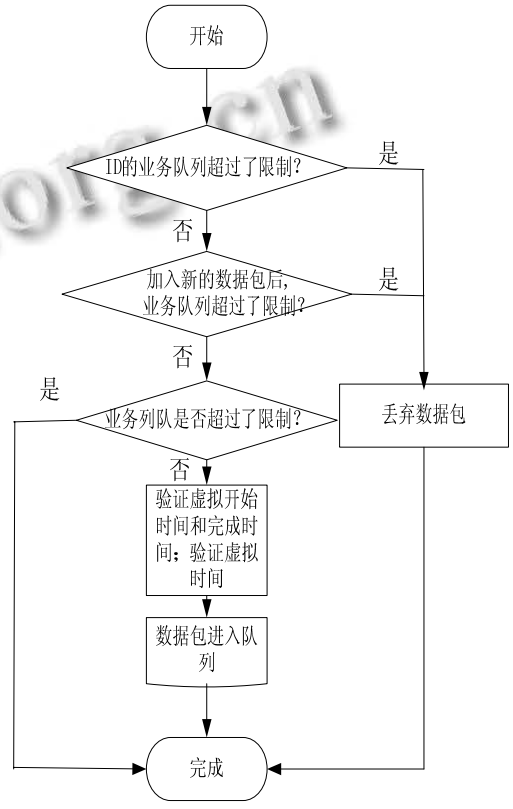

图 4 WF2Q+入队列流程图

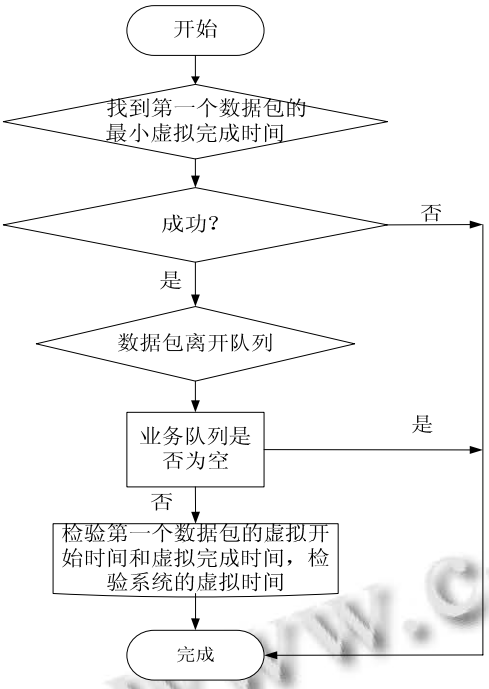

图 5　WF2Q+出队列流程图

2　各模块的算法设计与实现

2.1 控制状态分配

各个数据源的业务流所需求的服务大多都是关于带宽，延迟和延迟抖动等等的服务质量(QoS)参考值. 这些参考值都能够直观地反映网络对各个数据源提供的 QoS，但在本系统中，我们并不采用这些参考值作为系统的输入，而是选择各个数据源的权值作为系统的输入. 主要有以下 3 点原因：

1) 宽输入的特点：在一般的网络环境中，如果各个业务流的请求直接以所占的带宽量来作为系统的输入，那么就缺乏对整体系统资源总量的考虑. 由于网络环境异常复杂，带宽的分配并不是单个业务流的事情，带宽是由多个业务流所共通享有的. 因此，过大的带宽请求在网络繁忙时期显得无理而不可能实现，过小的带宽请求则在网络空闲时期造成总体带宽的浪费.

2) 据包延迟输入的特点：数据包延迟主要是由传播延迟和中间节点处理时间所影响的. 而传播延迟通常只占非常少的一部分，而且现今的路由器芯片的处理速度也非常高，包的解封和重新封装所占用的时间非常短，对数据包的延迟影响很小. 这就是说数据包的延迟主要受到中间节点的排队时间影响. 由于网络当中瓶颈链路的出现，中间节点的排队时间越长，就会严重影响数据包的延迟. 换个角度说，每个业务流所分得的带宽直接影响其数据包的延迟. 但是如果要统计数据包的延迟，那么从信息的反馈角度来看，使得系统带有更大的滞后性，这对于提高系统的实时性，提高系统的调度效率不利.

3）据包延迟抖动输入的特点：根据第(2)点的分析，我们不难看出数据包的延迟抖动，也可以看成是每个业务流所分得的带宽的稳定程度. 同样地，统计数据包的延迟抖动也会造成对系统的滞后影响. 而且，统计数据包的延迟抖动也会使中间节点的运算量大大提高，影响转发的效率.

综上所述，无论是采用直接的带宽请求，还是数据包的延迟请求或者延迟抖动的要求，都不利于提高系统的实时性和鲁棒性. 因此，我们需要一个既能够反映各个数据源的业务要求，同时能够尽可能不影响系统的实时性和鲁棒性，或者说能把影响降到最低的数值，作为系统的输入. 因此，我们选择了各个数据源的权值作为此反馈控制系统的输入. 设定各个数据源的权值，也就相当于设定了各个数据源的优先级. 以此为输入，我们就可以根据网络的实际情况，对不同优先级的数据源给予相应的服务，在网络繁忙时期对各个数据源提供有限服务，在空闲时期则提供尽可能多的服务，这样也能够同时兼顾系统的公平性.

2.2 各服务量统计器和总服务量统计器的设计

从控制框图中，各服务量统计器通过公式(1) 对各个数据源的实际服务量进行累计.

$$Q_i^k{}' = Q_i^{k-1}{}' + L_i^k \tag{1}$$

其中 $Q_i^k{}'$ 和 Q_i^{k-1} 分别为数据源 i 的成功转发第 k 和第 k-1 个数据包时的实际服务量. 假定当前转发的数据包为数据源 i 的第 k 个包，那么各服务量统计器将对数据源 i 的实际服务量进行更新. 从公式(1)我们可以看到，各服务量统计器每成功转发一个数据包时，只会针对该数据包所属的数据源 i 的实际服务量 $Q_i^k{}'$ 进行更新在本系统当中，总服务量统计器的功能就是根据把所有数据源从系统发出的数据量进行累计. 每当一个数据包成功从调度发送器中转发出去的时候，总服务量统计器就会根据一下公式(2)，对系统总服务量 Q_{sum} 进行累计.

$$Q_{sum}^k = Q_{sum}^{k-1} + L^k \tag{2}$$

上式中，Q_{sum}^{k-1} 表示在第(k 一 1)个数据包成功转发后的系统总服务量；L^k 表示现在刚转发成功的第 k 个数据包的大小；而 Q_{sum}^k 则表示第 k 个数据包成功转发后的系统总服务量.

2.3 预定服务量分配器的设计

所谓服务量，即数据源所发出的数据量. 在本控制系统当中，预定服务量分配器主要是对各个数据源的业务流所应该获得的服务量进行预估分配. 在系统运行的过程中，预定服务量分配器会根据各个数据源的业务请求所预设的权值作为其中一个考虑因素，然后综合由总服务量统计器反馈回来的系统当前总服务量 Q_{sum}，通过一定的机制，估算出各个业务流当前应当获得的服务量，也就是各个业务流对服务量的期望值. 然后，预定服务量分配器会把输出的各个业务流的服务量期望值再与系统反馈回来的各个业务流当前的实际服务量进行比较，并且把结果输入到权值调节器当中. 在本系统当中追求的服务量公平性可以用如下的公式(3)表示：

$$\frac{q_1}{w_1}=\frac{q_2}{w_2}=\ldots.=\frac{q_n}{w_n} \tag{3}$$

其中，q_i 和 w_i 分别表示业务流所获得的服务量和权值，i 为业务流的流号码. 从公式（3）可以看到，我们所追求的公平性就是要使各个客户能够对应其权值获得相应的服务量，也就是根据各个业务流的权值分配相应的带宽.

但是在实际的网络环境中，由于各种不确定因素影响，是不可能实现百分百的链路利用率的. 因此在设计预定服务量分配器的时候，不采用总带宽 B 乘以时间 t 这样直接的方法算出整个系统的总服务量. 由于系统的总链路利用率是变化的，为了提高系统的实时性，采用实时系统总服务量作为控制分配的反馈信息. 预定服务量分配器的内部算法可以用如下式来表示：

$$Q_i=Q_{sum}*w_i/W \tag{4}$$

其中，Q_{sum} 为系统反馈所得的系统实时总服务量，w 为当前队列不为空的业务流的权值总和. 预定服务量分配器根据公式(4)分别算出每个业务流的服务量期望值，然后根据如下公式(5)，对期望值和实际值进行比较.

$$\Delta Q_i=Q_i-Q_i' \tag{5}$$

其中，ΔQ_i,为 i 号业务流的服务量期望值和实际值之差，而 Q_i' 则为系统反馈所得的 i 号业务流的实际获得服务量. 最后，将得出的差值 ΔQ_i 输入到权值调节器当中.

2.4 预定服务量分配器的设计

权值调节就是指根据数据源的服务量期望值和实际值的差值 ΔQ_i，判断数据源是达到了期望服务量，还是超过了期望服务量，然后以原权值 w_i' 为基础，生成一个新的权值 w_i' 供调度发送器直接进行调度使用. 具体的权值修改方案我们可以通过下面的公式(6)来实现：

$$\begin{cases} w_i'=w_i*\alpha & \Delta Q_i>0 \\ w_i'=w_i & \Delta Q_i=0 \\ w_i'=w_i*\beta & \Delta Q_i<0 \end{cases} \tag{6}$$

从公式(6)可得，权值修改方案总共有三种情况：

(1) 当数据源 i 的服务量期望值和实际值的差值 ΔQ_i 大于零，也就是说数据源 i 所获得的实际服务量 Q_i' 不满足其期望值的时候，把其原来的权值 w_i' 以补偿系数 α，并以此作为其新的调度用权值 w_i'.

(2) 当数据源 i 的 ΔQ_i 等于零，也就是说数据源 i 所获得的实际服务量 Q_i' 刚好等于其期望值 Q_i 的时候，w_i' 只需要保持原有权值 w_i.

(3) 当数据源 i 的 ΔQ_i 小于零，也就是说数据源 i 所获得的实际服务量 Q_i' 超过了其期望值 Q_i 时，把原来的权值 w_i 乘以一个惩罚系数 β，并以此作为新的调度用权值.

2.5 调度发送器

本系统的调度发送器，采用最小合格虚拟完成时间优先的机制(Smallest E1igible virtual Finish time First，简称 SEFF 机制)来实现对各个数据源缓冲队列的调度发送. 所谓虚拟时间就是调度发送器通过各个数据源的权值和数据包的大小，估算出的数据包发送所需时间，而并非数据包实际发送所需的时间. 由于系统实际运行过程中还涉及到很多因素，因此实际上，数据包的发送时间会比系统估算出的虚拟时间大. 但是使用虚拟时间来对数据源的缓冲队列进行调度，并不会影响到系统的公平性. 下表 1 包含了本调度发送器所使用的一些量的数学符号及其解释,下面所提及

的时间均为虚拟时间.

表 1 调度发送器的数学符号

$V(a_i^k)$	数据源 i 第 k 个数据包到达系统的时间
S_i^k	数据源 i 第 k 个数据包开始发送的时间
F_i^k	数据源 i 第 k 个数据包完成发送的时间
W	当前缓冲队列不为空的数据源的原权值 w_i 的总和
L_i^k	数据源 i 第 k 个数据包的大小
w_i'	从权值调节器中获得的数据源 i 的调度用权值
$Qlength_i$	数据源 i 当前的缓冲队列长度
$misS_i^k$	当前各个数据源中开始发送时间最小的源的发送时间

本调度器的整个调度过程分成数据包进入缓冲队列和数据包的转发两大部分. 最初, 没有任何数据包发送的时候, 所有数据源的开始发送时间 S_i^k 和完成发送时间 F_i^k 都为零. 当有数据包发送到系统当中的时候, 调度发送器就开始对该数据包所属的数据源 i 进行如下两种情况的处理:

如果 $Qlength_i \neq 0$, 即当数据源 i 的缓冲队列不为空的时候, 数据包直接进入对应的缓冲区, 系统并不更新数据源的开始发送时间影和完成发送时间 S_i^k. 如果 $Qlength_i = 0$, 即当数据源 i 的缓冲队列为空的时候, 系统将会分别对开始发送时间 S_i^k, 完成发送时间 F_i^k 和系统时间 $V(a_i^k)$ 进行更新. 首先, 系统将根据如下公式:

$$S_i^k = MAX\{F_i^{k-1}, V(a_i^k)\} \quad (7)$$

对该数据源 i 的开始发送时间 S_i^k 进行更新. 当数据源 i 的第 k 个数据包发送到其缓冲队列的时候, 系统将直接把数据源 i 第 $(k-1)$ 个数据包的发送完成时间 F_i^{k-1} 设为 S_i^k, 如果数据源之前并没有发送过数据包, 即 k=l 的时候, 系统将把当前的系统时间 $V(a_i^k)$ 设置为 S_i^k; 或者当 F_i^{k-1} 比 $V(a_i^k)$ 小的时候, 我们也会把 $V(a_i^k)$ 设置为 S_i^k. 然后, 系统将根据公式

$$F_i^k = S_i^k + L_i^k / w_i' \quad (8)$$

把数据源 i 的调度用权值直接看成是所分配到的带宽 w_i', 这样来估计系统按照 w_i' 分配所得的带宽, 完成数据包 k 的发送所需要的时间. 最后, 系统将根据公式

$$V(a_i^k) = MAX\{misS_i^k, V(a_i^{k-1})\} \quad (9)$$

选择最小发送时间和原系统时间中较大的一个作为新的系统时间 $V(a_i^k)$, 就是说系统必需确保系统时间 $V(a_i^k)$ 大于或者等于最小发送时间 mis.

系统中调度发送器使用的是 SEFF 机制. 所谓 SEFF 机制, 就是说在系统实施调度的过程中, 将会优先发送完成时间 F_i^k 最短的数据源的数据包. 然后, 系统将会进行对应数据的更新, 数据的更新方案分两种:

1) 发完成后缓冲队列为空

假设系统现在选择了数据源 i 的数据包进行转发, 但是数据源 i 的缓冲队列中只剩下唯一的一个数据包 k 了, 当数据包被系统转发后, 此缓冲队列就会变成空的. 在这种情况下, 系统不对该数据源 i 的开始发送时间 S_i^k 和完成发送时间 F_i^k 进行更新, 而只对系统时间 $V(a_i^k)$ 进行更新. 此时, 我们按照公式(10)更新系统时间.

$$V(a_i^k) = MAX\{\min S_i^k, V(a_i^k) + L_i^k / W\} \quad (10)$$

注意: 当中各个量的上标 k 都只是代数, 并不是指同: 一个数据包, 而下标也并非指同一个数据流. 两个所指的也分别是两个不同时刻的系统时间. 当系统完成对系统时间的更新后, 就会继续从缓冲区中选择下一个数据包发送. 而缓冲队列变为空状态的数据源 i 则继续等待新的数据包的到来, 然后将进行如上面提到的第一部分数据包进入缓冲队列的第二种情况, 即的情况, 对开始发送时间, 完成发送时间和系统时间行更新. 与上面一种情况刚好相反, 数据源的缓冲队列有不止一个数据包, 那么当数据包被转发之后, 系统就必需对数据源的开始发送时间和完成发送时间进行更新了. 其开始发送时间的更新方法公式（11）所示.

$$S_i^k = F_i^{k-1} \quad (11)$$

完成第(k-1)个数据包发送时间可以设置为第 k 个数据包发送起始时间. 而完成发送时间 F_i^k 的更新方法则依公式（8）进行, $V(a_i^k)$ 则按公式（10）处理.

3 仿真实验

我们用 NS-2 作为实现算法和测试性能的仿真平台. 作为一个开源软件,NS-2 已经应用广泛,成为一个主要的通信和计算机网络的研究仿真平台. 因为 NS-2

采用了开放式体系结构的大规模数据库的协议,它已经被广泛的应用在局域网,广域网,自组织网络、移动网络和卫星网络仿真.

实验中采用个 n TCP 业务流和个 m UDP 业务流来模拟实际应用过程中网络中间节点面对的调度问题, 检验系统对不同传输协议的公平性. 所有业务流与路由器 A 之间的链路容量为 1Mbps, 延迟为 20ms, 瓶颈链路位于路由器 h 和路由器 B 之间, 容量为 5Mbps, 延迟为 lOms; 路由器 A 和路由器 B 之间的链接使用本算法, 其余的均实用 DropTail; 每个客户端发出的业务流的包大小基本不相同.

使用以下一系列的实验来说明本算法对系统的带宽分配能力更强. 本组实验通过预设两种算法的参数, 设置一部分客户端先启动, 另一部分则较迟启动, 观察两种算法的短期公平性.

设置 5 个使用 FTP 服务的客户端(由于 FTP 服务实用 TCP 传输协议, 以下简称 TCP 业务流)和 2 个用 CBR 流量发生器模拟的 UDP 业务流, 分别设置为 0—6 号业务流, 其中 O 一 4 号为 TCP 业务流, 5、6 号为 UDP 业务流(如图 5—5). 其中 0—3 号四个 TCP 业务流为先启动组, 先启动组从 0 秒开始传输数据, 4—6 号业务流为后启动组, 启动时间设置为 10 秒. 所有业务流工作到实验结束时间 90 秒为止. 而路由器 A 分别使用 WF2Q+和本算法进行实验. 其中, 本算法的补偿系数设置为 10, 惩罚系数设置为 0.5. 通过对实验进行数据分析, 比较两种算法在实现短期公平性方面的性能优劣. 首先, 根据公式(22), 求出后启动组在任意时刻的期望带宽.

$$B_a(t) = B(t) * \frac{w_a}{W} \tag{12}$$

其中, 为 t 时刻的后启动组的期望带宽, 为 t 时刻实际总带宽, 为该时刻后启动组的各个业务流的权值之和, 而 W 则为该时刻正在传输数据的各个业务流的权值总和. 另外, 通过公式(13), 可以求出后启动组任意时刻的瞬时带宽与其当时期望值的比值.

$$\phi(t) = \frac{B_a'(t)}{B_a(t)} \tag{13}$$

其中, 为 t 时刻后启动组各个业务流的带宽总和. 通过比值, 我们可以看出后启动组的业务流获得的实际带宽与其期望值相差多少. 通过上面三组对比实验, 得出如下表 2 的一数据实验数据.

表 2 不同情况下反馈调度方法和 WF2Q+的性能指标

后启组动时间	路由使用算法	后启动组瞬时带宽与期望带宽的比值		
		40s	60s	80s
10s	WF2Q+	80.89%	87.88%	91.33%
	反馈调度	97.55%	100.01%	100.78%
15s	WF2Q+	69.02%	80.49%	85.85%
	反馈调度	88.72%	97.97%	100.05%
20s	WF2Q+	61.14%	78.08%	85.75%
	反馈调度	73.33%	93.28%	99.93%

上表 2 数据显示, 后启动组不论是从 10s、15s 还是 20s 的时候开始传输数据, 本算法在 40s、60s 和 80s 三个采样时间当中, 其获得的瞬时带宽与期望带宽的比值, 都比使用 WF2Q+更加接近 100%. 在这种情况下, 使用本算法能够为后启动组获得期望服务量提供更好的保证, 实现更好的短期公平性. 使用本算法能够获得更公平的网络服务, 更好的 QoS.

4 结 语

本文提出了一种基于短时公平性的分组反馈调度算法. 在传统的 WF2Q+算法的基础上, 结合反馈控制理论, 设计出基于短时公平系的业务管理模型和反馈控制模型. 通过加入反馈控制环节, 改善了原有 WF2Q+算法中对新加入客户端无法很好地实现短时公平性的缺陷, 使得系统能够根据各个客户端所获得的实时服务量, 迅速调整各个客户端的调度权值, 从而补偿获得服务量过小的客户端和惩罚获得服务量过多的客户端.

参考文献

1 Lei Y, Geir E. Global stability of internet congestion controllers with heterogeneous delays. IEEE Trans. on Networking, 2006, 14(3): 579-583 .

2 Peet M, Lall S. Global stability analysis of a nonlinear model

of internet congestion control with delay. IEEE Trans. on Auto. Cont., 2007, 52(3): 553-560.

3 Sichitiu ML, Bauer PH. Asymptotic stability of Congestion Control Systems With Multiple sources. IEEE Trans. on Auto. Cont., 2006, 51(2): 292-298.

4 La RJ, Ranjan P. Asymptotic Stabilityof Rate control system with communication delays. IEEE Trans. on Auto. Cont., 2007, 52(10): 1920-1926.

5 Wang Z, Paganini F. Boundedness and global stability of nonlinear congestion control with delays. IEEE Trans. on Auto. Cont., 2006, 51(9): 1514-1520.

6 Hitay PFQ. On the design of AQM supporting TCP flows using robust control theory. IEEE Trans. Auto. Cont. 2004, 49(6): 1031-1036.

7 Deb S, Srikant R. Global stability of congestion controllers for the Internet. IEEE Trans. Auto. Control, 2003,48(6): 1055–1060.

8 Ranjan P, La R, Abed E. Global stability conditions for rate control with arbitrary communication delays. IEEE/ACM Trans. on Networking, 2006, 14(1): 94–107.

9 Chiu DM, Jain R. Analysis of the increase and decrease algorithms for congestion avoidance in computer networks. Computer Networks and ISDN Systems, 1989, 17(1): 1-14.

10 Kelly F, Maulloo A, Tan D. Rate control in communication networks: Shadow prices, proportional fairness and stability. J. Oper. Res. Soc.,1998, 49: 237-252.

11 Tian Y, Yang H. Stability of the Internet congestion control with diverse delays. Automatica, 2004, 40: 1533-1541.

12 Liu S, Basar T, Srikant R. Exponential-RED: A stabilizing AQM scheme for low-and-high-speed TCP protocols. IEEE/ACM Trans. Networking, 2005, 13(5): 1068–1081.