高性能路由器分组调度算法研究*

江 勇, 吴建平, 徐明伟

(清华大学 计算机科学与技术系 网络研究所,北京 100084)
E-mail: jyong@csnet1.cs.tsinghua.edu.cn
http://netlab.cs.tsinghua.edu.cn

摘要: Internet 同时面临着两个问题:更快的交换路由结构和引入服务质量(QoS)保证.每个问题都可以独立解决.高性能路由器可以用输入缓冲的交叉开关(crossbar)代替共享内存来获得更快的速度;QoS 能够通过分组公平排队算法 PFQ(packet fair queuing)来得到.然而到目前为止,这两个问题的解决还是互斥的——所有的分组公平排队算法研究都需要路由器采用输出排队或者集中式共享内存.基于输入输出结合排队 CIOQ(combined input output queuing)结构,设计和实现了一种分组调度算法 DF^{2}Q(distributed feedback fair queuing).该调度算法最重要的特征是引入了反馈机制.分析并讨论了 DF^{2}Q 的性能.实验结果表明,它能够很好地避免内部拥塞和提高资源利用效率.

关 键 词: 分组公平排队;输入输出结合排队;反馈
中图法分类号: TP393 文献标识码: A

对于旨在提供服务质量(quality of service)的分组调度算法已经进行了很多的研究工作,其中绝大多数采用的是输出排队机制.对于输出排队,每到来一个分组,它就被立即放置到输出端的队列中,在那里等待调度输出.这种输出排队的方法,可以比较方便地控制分组的时延,从而提供服务质量的保证.但是随着端口速率在2.5G(OC-48)以上且端口数较多的高性能路由器的出现,单纯的输出排队机制已不能满足网络技术发展的需要.例如,对于一个 N 端口的高性能路由器,其内部交换结构(crossbar)和存储器都需要工作在 N 倍线速,即需要加速比为 N,同时要求输出端的控制逻辑必须运行在一个极高的速率,这就使得输出排队方式的可扩展性很差.为了解决这些问题,大部分的高性能路由器(研究[1]和商用[2,3])都选择了带输入排队的体系结构.通过在输入点加入缓冲,路由器的内部交换和存储访问可以以一个较低的速度进行.这样输入排队可以很容易地解决可扩展性问题.

但如何把输入排队和输出排队结合起来就成了我们要考虑的问题.文献[4]从理论上证明了这种CIOQ(combined input output queuing)排队方式在加速比为 2 的情况下就足以模拟任何输出排队的输出效果.因此,可以在输入端和输出端采取适当的分组调度算法来对不同的应用提供服务质量(QoS)保证.

本文提出了一种在分布式的 CIOQ 体系结构下的调度算法 DF^{2}Q(distributed feedback fair queuing).该调度算法引入了一种输出端到输入端的反馈机制,主要目的是为了避免拥塞和输出端的链头阻塞 HOL(head of line).我们的分组调度算法提供和输出端公平排队近似的服务质量保证,同时,这种分布式 CIOQ 体系结构不需要全局的同步即能实现高速流水,保证了路由器的转发性能.我们引入的反馈机制通过在输出端定期检测拥塞状况,然后按照一定的预测规则在将要发生拥塞时向相应的输入端发送反馈信息,输入端对这种信息做出反应,

* 收稿日期: 2000-07-05; 修改日期: 2000-10-16
基金项目: 国家自然科学基金资助项目(69682002;69725003);国家 863 高科技发展计划资助项目(863-306-2D-07-01)
作者简介: 江勇(1975－),男,重庆人,博士生,主要研究领域为计算机网络体系结构,高性能交换结构,调度算法;吴建平(1953－),男,山西太原人,博士,教授,博士生导师,主要研究领域为计算机网络体系结构,计算机网络协议测试,形式化技术;徐明伟(1971－),男,辽宁朝阳人,博士,副教授,主要研究领域为计算机网络性能分析和测试.

以减慢向该输出端的发送速度,从而缓解或者避免拥塞.

本文第 1 节简单介绍一些分组调度研究的相关背景.第 2 节介绍高性能路由器的分布式体系结构和在分组转发方面的功能要求.第 3 节讨论我们采用的调度算法和反馈机制.第 4 节为性能测试与分析.第 5 节对这种反馈机制进行进一步的讨论.第 6 节总结了全文.

1 研究背景

1.1 分组调度

为支持服务质量(QoS)保证,人们提出了很多基于会话的分组公平排队算法 PFQ(packet fair queuing)[5~8].而这些算法大部分都采用的是输出排队机制,正如我们前面提到的,输出排队的可扩展性差,在端口数较多且线速很快的高性能路由器中就不再适合了.因此,人们考虑采用输入排队方式.然而输入排队的引入带来一个问题,以前只在输出端存在竞争,而现在在输入、输出均存在着竞争——来自同一个输入的分组可能去向不同的输出端,而来自不同输入端的分组也可能去向同一个输出端.若输入调度为 FIFO 在输入端就不存在竞争,但会引入链头阻塞(HOL)[9]:若队列头的分组因为竞争输出被阻塞,则在同一个输入队列去向另外空闲输出端的分组也不能被转发.如何解决在输入排队的路由器中的竞争问题是我们关注的重点,而目前大部分研究工作着眼于提高交换结构的吞吐率[10~13].而如何在输入缓冲和中加速比的情况下提供 QoS 研究的较少.

1.2 分组公平排队

在当前分组调度算法的研究方面,各种文献中提出的算法主要分为 3 大类:连续工作型算法(work-conserving)、断续工作型(non-work-conserving)算法和层次型算法[14].关于调度算法的综述可以参见文献[14].

分组排队的最初目的是为了避免拥塞,但合理的分组调度算法的采用还可以对服务质量提供支持.目前,在商业化的路由器中大多支持的是分组公平排队类的算法.

分组公平排队算法是模拟广义处理器共享(general processor sharing,简称 GPS)[15]的一类分组调度算法.对于一个有 N 个会话的 GPS 系统,每个会话 i 所占的份额由一个正实数 ϕ_i 所标识.在每一个时间间隔,如果系统中有 $M(M\le N)$个非空的队列,那么服务器按照它们各自的份额的比例同时为这 M 个队列头部的 M 个分组提供服务.会话 i 在时间间隔$[t_1,t_2]$内得到的服务 $W_i(t_1,t_2)$表示如下:

$$W_i(t_1,t_2)\ge \frac{\Phi_i}{\sum_{j\in B(t_1,t_2)}\Phi_j}r(t_1-t_2),$$

其中 $B(t_1,t_2)$为在时间间隔$[t_1,t_2]$内有非空队列的会话的集合.

分组公平排队算法都需要维护一个系统虚拟时钟 $V(t)$,同时为每一个会话 i 有一个虚拟起始时间 $S_i(t)$和虚拟结束时间 $F_i(t)$.从直观上可以这样理解,$V(t)$代表每个会话在时刻 t 时应该已经接受到的服务时间;$S_i(t)$表示会话 i 在时刻 t 时已经接受的服务时间,它通常也被称作会话 i 的虚拟时间而记作 $V_i(t)$;$F_i(t)$为 $S_i(t)$加上服务它的队列头的分组所需的时间.所有 PFQ 算法的目标就是为了尽可能地减小所有 $V_i(t)$和 $V(t)$之间的差距.$S_i(t)$和 $F_i(t)$的计算和更新采用如下公式:

$$S_i(t)=\begin{cases}\max(V(t),S_i(t)), & \text{会话}i\text{激活时}\\ S_i(t)+\dfrac{L_i^k}{r_i}, & \text{分组}p_j^k\text{结束服务时}\end{cases},\tag{1}$$

$$F_i(t)=S_i(t)+\frac{L_i^k}{r_i}.\tag{2}$$

各种公平队列算法的区别就在于系统虚拟时间 $V(t)$的计算和分组选择策略的不同上.对系统虚拟时间函数的计算,如将其设置为正在享受服务分组的起始时间[5]、正在服务分组的结束时间[16]、系统中所有非空队列头的分组的起始时间的最小值[6]等等.分组选择策略有最小起始时间优先[5]、最小结束时间优先[16]以及最小合格分组的结束时间优先[6]等.不同的系统虚拟时间 $V(t)$的计算也在很大程度上决定了这种分组公平队列算法的复

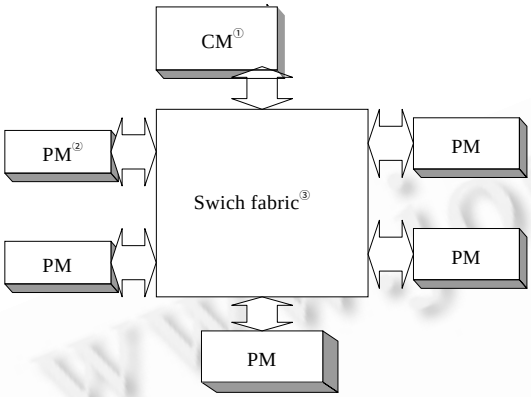
杂性和调度性能.

下面我们所介绍的用于高性能路由器的分布式反馈分组调度算法 DF^2Q 就是一类分组公平队列算法.在输入端和输出端分别有多个公平队列,中间通过交换结构相连接.

2 路由器的分布式体系结构

2.1 硬件体系结构

我们的高性能路由器在硬件上采取的是主从式的多处理器分布式体系结构,一个主处理器模块,多个从处理器模块.这种体系结构如图 1 所示.主从处理器模块的功能如下所述:



①中央处理器,②从处理器,③交换结构.

Fig.1 High performance architecture of router

图 1 高性能路由器硬件体系结构

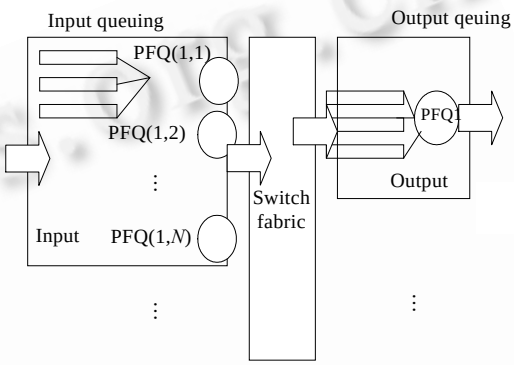


Fig.2 Architecture of packet scheduler

图 2 分组调度的逻辑结构

(1) 主处理器模块 CM(central module)

- 处理路由协议包,维护全局路由表
- 保持局部路由表与全局路由表同步
- 完成对路由器与网络的操作管理
- 维护全局数据库

(2) 从处理器模块 PM(peripheral module)

- 使用局部路由表完成 IP 包的转发
- 使用安全数据库完成 IP 包的过滤
- 支持多种网络接口协议
- 对本处理器相连接口单元的支持

2.2 转发调度机制的逻辑结构

采用交换结构(crossbar)的分布式的体系结构的目的就是要支持数据的高速转发以达到高性能的要求.

不同接口之间的数据转发通过交换结构来完成,转发调度机制的逻辑结构如图 2 所示.我们采取了 CIOQ 的体系结构,在输入接口和输出接口都有基于每个流的排队.在输入端口的每个流排队再按照输出端口的不同组织成多个虚拟输出队列 VOQ(virtual output queue),这样可以有效地解决 HOL 问题.

对于 N 个端口而言,在每个输入端都要有 N 个调度器,每个调度器对应一个 VOQ,它负责调度转发该 VOQ 中的若干个流到相应的输出端口.各个调度器都采用一致的分组调度算法以保证服务的公平性.交换机构对这 $N\times N$ 个调度器的调度结果再进行选择和调度,它寻找合适的 N 个输入端和 N 个输出端的匹配,把它们送入各个输出端的每个流队列中.输出端的调度就是通常的输出队列调度,每个输出端有一个调度器负责调度分组输出.

我们的各个调度器采用的都是一种分组公平队列调度算法.输入端 i 的 N 个调度器,记作 PFQ(i,j),$1\le i\le N$, $1\le j\le N$;输出端 j 的一个调度器记作 PFQ j,$1\le j\le N$.

3 分组调度算法和反馈机制

3.1 分组调度算法

分组公平调度算法如前面第1节所述.PFQ(i, j)的系统虚拟时间记作 $V_{i,j}(t)$,PFQ j 的系统虚拟时间记作 $V_j(t)$.每一个流 f 要求分配的速率记为 Rf,路由器实际分配的速率记为 rf.

$$V_{i,j}(t)=\max\{v(\tau)+t-\tau,\min_{n\in B(t)}S_{i,j,n}(t)\}, \tag{3}$$

其中 τ 为 t 时刻前最后一个分组调度输出的时间,$B(t)$ 为 t 时刻有非空队列的流的集合.

$$S_{i,j,f}(t)=\begin{cases}F_{i,j}(t), & \text{分组}p_f^k\text{服务结束时},\\ \max\{V_{i,j}(t),F_{i,j,f}(t)\}, & \text{分组到来时流}f\text{队列为空},\end{cases} \tag{4}$$

$$F_{i,j,f}(t)=S_{i,j,f}(t)+L_f^k/r_f. \tag{5}$$

在最初流的接入控制中,rf=Rf;但是之后 rf 会随着反馈信息而有所变化,具体变化情况详见后文的分析.每一个流都有一个起始时间和结束时间标记,每个分组公平队列调度器根据这些标记来进行分组的调度.我们实现的分组公平调度算法是 WF^{2}Q+[6].对调度器 PFQ(i,j),其虚拟时间函数按式(3)计算;$V_{i,j}(0)=0$,$F_{i,j,f}(0)=0$;各流的起始时间标记和结束时间标记按照公式(4)和(5)计算.在算法实现时,初始 $V_{i,j}(t)=0$,各流的起始时间标记均为 0.虚拟时间的更新和各流的起始时间标记只在下面两种情况下计算:一种是流 f 的分组到来时流 f 队列为空,此时,按照式(4)计算新的 $S_{i,j,f}(t)$,所用的 $V_{i,j}(t)$ 是按照式(3)在原来值的基础上计算的新值,从而可以计算出其结束时间标记;另一种情况是在选取调度流 f 的一个分组 pfk 后,按照式(4)和式(5)重新计算该流的起始时间标记,如果流队列中还有分组,则计算新的结束时间标记.之后可以选择调度有最小结束时间标记的流的分组.对输出端的分组调度器 PFQj,也采用相同的算法进行调度输出.

用于交换结构的调度算法,目前也有很多研究成果.文献[17]中提出一种称为 JPM(joined preferred matching)的交换结构的调度算法,可以配合输入队列的分组公平调度算法工作,有效地模拟输出队列的性能.文献[18]中提出一种启发式的模拟跟踪流策略的交换结构调度算法,通过检测 Critical Node 和结点间的匹配来提供对时延等方面的服务质量的保证.这些算法都可以根据实际情况的需求应用于我们的交换机构中,并与前后的输入输出队列的分组公平队列调度算法一起配合起来协调工作.考虑到我们选取的 Crossbar 调度芯片等因素,我们在实际中采用的是 JPM 的调度算法.

3.2 反馈机制

输出端的拥塞会造成大量分组丢失,进而导致大量的分组重传,加重系统负担.采用适当的反馈机制可以有效地避免这种情况的发生.在我们的高性能路由器上，我们设计了一种定时检测的基于每流预测的拥塞反馈机制.

这种反馈机制的原理如下:对某一个输出端 i 而言,当输入到这个输出端的流 f 的速率(记作 rfin_i)大于输出的速率(记作 rfout_i,)时,输出端的队列中的元素呈增长趋势,拥塞有可能发生.可以根据此时的队列输入速率和输出速率估计该流的队列将要达到饱和的时间 tf_full_i.如果这个时间小于某个预定的阈值,则拥塞发生的概率将非常大,需要向这个流来自的相应输入端的分组调度器发送反馈信息,报告拥塞的预测状况.输入端的分组调度器根据这个信息,适当地调整可能拥塞的流的实际分配速率 rf,减缓向那个输出端口调度输出的速率,避免拥塞的发生.

这种反馈机制的实现如下:设定时检测的定时间隔为 T,拥塞时间阈值为 $t_{\max}$ 和 $t_{\min}$.

(1) 输出端 i 的流 f 的输入速率 rfin_i 的估计.输入速率实际上是 n 个 T 时间的平均输入速率.每个流的输出队列需要记录此前 n 个 T 时间中输入的分组的总长度,这个值随着每一个分组的入队以及每个 T 间隔的到来而不断更新.

(2) 输出端 i 的流 f 的输出速率 rfout_i 的估计.与上面的输入速率类似,它也是 n 个 T 时间的平均输出速率.输出队列记录此前 n 个 T 时间的输出该流的分组总长度,这个值随着每一个分组的出队以及每个 T 间隔的到来而不断更新.

(3) 拥塞的预测和反馈信息的发送.rfin_i>rfout_i,即当有可能发生拥塞时,估计队列将要饱和的时间 $tf\_$

full_i.用下面的公式计算估计值:

$$t_{f\_full\_i}=\frac{\text{队列空闲元素数目}}{r_{fin\_i}-r_{fout\_i}}.$$

当 $tf\_full\_i>t_{max}$ 时,认为拥塞发生概率很小,不发反馈信息.当 $t_{min}<tf\_full\_i\le t_{max}$ 时,预测为一般拥塞,发送一般反馈信息.当 $tfull\_i\le t_{min}$ 时,认为拥塞即将发生,发送紧急反馈信息.需要发送反馈信息时,输出端向流 f 的输入队列所在的输入端口发送反馈信息.

(4) 输入端对反馈信息的处理.

对于一般的反馈信息,减小向对应输出端口输出的流 f 的实际分配速率 rf,$rf\leftarrow\alpha rf$,减小因子 $\alpha=rf\text{out}\_i/rf\text{in}\_i$,当收到同一个输出端口发来的一般反馈信息超过 m 次时,不再理会.对于紧急反馈信息,需要较大幅度的减小实际分配速率 rf,$rf\leftarrow\beta rf$,减小因子 $\beta=rf\text{out}\_i/\omega rf\text{in}\_i$,$\omega>1$.当 k 个定时间隔 T 内没有收到反馈信息时,实际分配速率应逐渐回升,$rf\leftarrow\gamma rf$,回升因子 $\gamma>1$,直至回升到该流的请求分配速率 Rf 为止.

(5) 若干个常数的选定.

本反馈机制中需要用到不少的常数:

定时间隔 T,预测拥塞发生的时间阈值 t_{max} 和 t_{min}.在紧急反馈信息处理中,对实际分配速率的减小因子 β 中的大于 1 的常数 ω.用于输入端响应的有效的一般反馈信息次数 m,用于输入端分配速率回升判断的 k 以及回升因子 γ.这些常数对于反馈机制的性能有很大的影响.对于这些常数的取值,我们采取了先取经验值然后再用实验辅佐的方法来确定.我们选取 T 为在无输入时饱和的输出队列清空所需时间的 1/8.用于输入速率估计值的 n 取值为 4,这样输入速率的计算就是基于饱和输出队列清空所需时间的二分之一.

4 性能测试与分析

4.1 实验环境简介

我们的实验环境如图 3 所示,处于中央的是我们自行开发的基于 Mortorola 的 MPC750 的高性能路由器原型系统,带有多个 100Mbps 和两个 1000Mbps 的以太网端口,另外 4 台 PC 机作为输入和输出结点,通过 100Mbps 的以太网络和中央路由器相连.

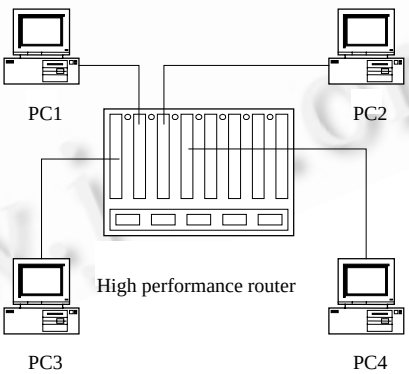


Fig.3 Test environment

图 3 测试环境

4.2 实验结果及性能分析

在实验中,我们为每个结点都建立了需要转发到其余 3 个结点上的若干个流.其中我们着重观察结点 1 的情况.下面是各个从结点需要转发到从结点 1 的几组流,流的参数见表 1.

Table 1 Flows in the test

表 1 实验中的流

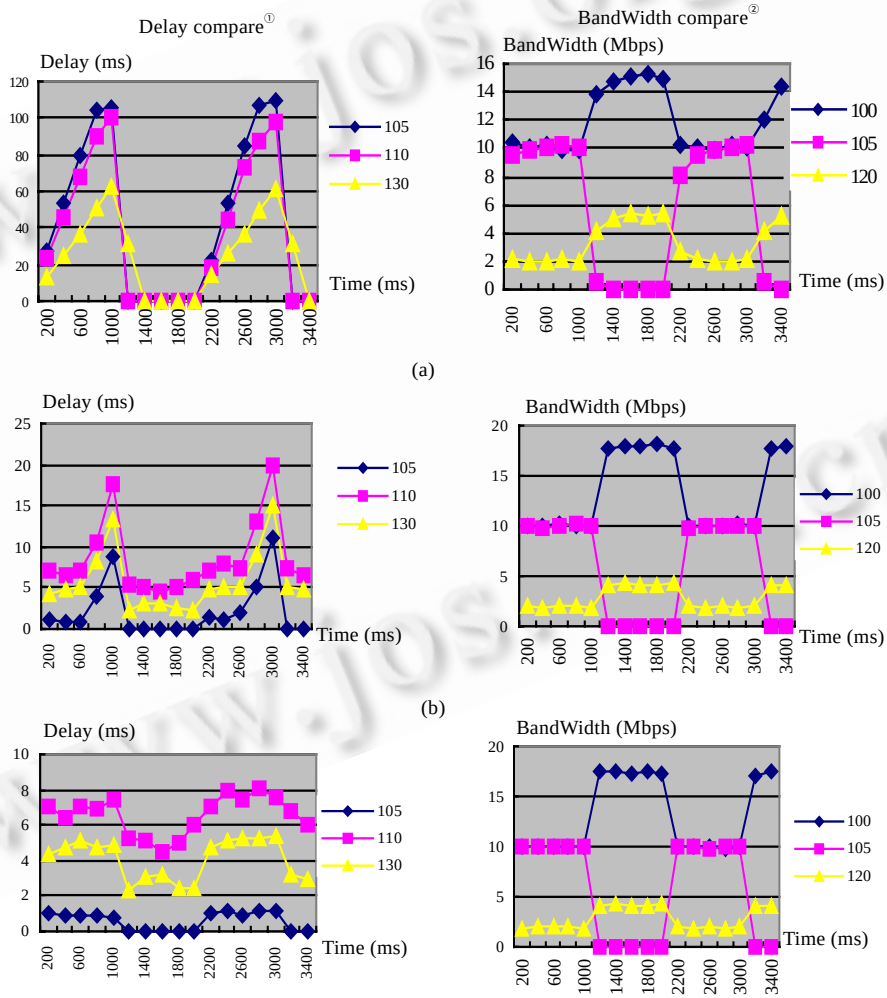
FlowID①	Flow type②	Reserved rate③ (bps)	Arrival rate④ (bps)	Packet length⑤ (Bytes)	Delay⑥	Input⑦
100~103	CBR	10M	20M	1 200	N/A	3
105~108	On/Off	10M	10M	1 200	1ms	4
110~113	CBR	1M	1M	1 200	10ms	3
120~123	Poisson	2M	6M	520	N/A	2
130~133	CBR	100k	100k	520	8ms	2

①流 ID,②流类型,③预留速率,④到达速率,⑤分组长度,⑥延迟,⑦入节点.

可见,流 100~103 和流 120~123 的实际发送速率大于它们的保留速率.按照这种速率发送下去,必将引起它们在输出端的拥塞.实验结果也证明了这一点.

由于篇幅限制,我们只给出了部分的测试结果.测试结果如图 4 所示.

图 4(a)和图 4(b)是无反馈的情况.,图 4(a)为输出缓冲队列较小,仅为 100Kbyte 时的情况;图 4(b)是在输出缓冲队列较大,为 1Mbytes 的情况;图 4(c)是带反馈机制的情况



①延迟比较,②带宽比较.

Fig.4 Performance test result

图 4 性能测试结果

从图 4 中可以看出,我们的分布式公平排队算法能够很好地模拟输出排队的服务质量要求,同时我们看到加入反馈机制后对带宽分配基本没有影响.但是从图 4 中的延迟比较可以看到,加入反馈机制后的分组转发延迟在系统资源发生冲突时也不会增大,而对于无反馈机制的分组调度策略在系统资源紧张时不可避免地会发生拥塞现象,严重影响转发性能,导致分组丢失率上升、网络性能下降.当然,增大系统资源(缓冲队列长度)可以部分避免拥塞,但系统资源不能无限制地增加,而且也影响系统的可扩展性.这其中的原因主要在于反馈机制的使用使得拥塞发生的危险有所减少,从而避免了拥塞发生时产生的额外开销,也就减小了时延.因此带反馈的分组公平调度算法性能较单纯的分组公平调度算法性能要有所提高.另外,我们从图中也可以看出加入反馈机制后增加的额外开销很小,不会影响整体性能.

5 进一步的讨论

在我们的分布式体系结构中,分组的调度输出主要分为 3 步:(1) 输入端对输入的流按照输出端口的不同,N 个分组调度器(假设有 N 个输出端口)并行工作,采用相同的分组公平排队调度算法计算和选取分组;(2) 交换机构的输入端首先对各分组调度器的输出分组进行排序,寻找最佳的输入输出匹配,将分组送到输出端的待调度队列中;(3) 输出端的调度器即一般的输出队列调度器,它计算选择合适的分组真正地输出到输出线路上.

在加入反馈机制的情况下,收到反馈信息的输入端口的分组调度器减小可能拥塞的流 f 的实际分配速率 $rf(rf=\alpha rf=(rf\text{out}\_i/rf\text{in}\_i)rf,\alpha=(rf\text{out}\_i/rf\text{in}\_i)<1)$,然后重新根据该流的起始时间标记 $S_{i,j,f}(t)$和式(4)来计算新的结束时间标记 $F_{i,j,f}(t)$,重新对各流的结束时间标记进行排序.从式(4)和式(5)可以看出,在减小实际分配速率 rf 后,流 f 的结束时间标记 $F_{i,j,f}(t)$和调度该流分组后新的起始时间标记 $S_{i,j,f}(t)$都将增大.分组调度器 PFQ(i, j)选择结束时间标记最小的流的分组,这样,流 f 被选择的概率将减小,从而实际上减缓了发送到那个输出端口的流 f 的分组的速度,减小了流 f 发生拥塞的可能性.

反馈机制的好处在于可以有效地避免拥塞,提高资源利用率.另外,我们的这种反馈机制是基于流的检测机制,输出端口在预测到某流 f 的拥塞要发生时,仅向流 f 存在的输入端口发反馈信息,有很强的针对性,输入端调度器仅减小流 f 的实际分配速率.这样就可以很好地隔离不同流之间的影响,避免坏行为流对好行为流的不利影响,也更好地支持了公平队列调度算法的公平性.

6 总 结

路由器系统的性能主要体现在数据的转发效率上,拥塞的发生必将极大地降低转发性能.本文结合一种分布式的路由器体系结构,讨论了应用于这种环境下的分组调度算法,主要贡献是提出了用于这种体系结构中的一种基于流的预测和反馈的拥塞控制算法.这种拥塞控制算法在定时检测的基础上,对每个流的拥塞发生的可能性进行预测,根据预测的结果向该流的输入端发送反馈信息,输入端采取一定的措施来减缓该流的实际分配速率,减小向预测可能会拥塞的输出端的发送速度,有效地实现了拥塞避免.

References:

[1] McKeown, N., Izzard, M., Mekkittikul, A., *et al*. The tiny tera: a packet switch core. IEEE Micro, 1997,17(1):26~33.

[2] Digital Equipment Corporation. GIGAswitch. 1997, http://www.networks.digital.com.

[3] Ascend Communications. GRF family of switches. 1998, http://www.ascend.com.

[4] Chuang, S., Goel, A., McKeown, N., *et al*. Matching output queueing with a combined input/output-queued switch. IEEE Selected Areas in Communications, 1999,17(6):1030~1039.

[5] Goyal, P., Vin, H.M., Chen, H. Start-Time fair queuing: a scheduling algorithm for integrated services. IEEE/ACM Transactions on Networking, 1997,5(5):690~704.

[6] Bennett, J.C.R., Zhang, H. WF^2Q: worst-case fair weighted fair queuing. In: Tatsuya, Suda, ed., Proceedings of the Infocom'96. Jersey: IEEE Piscataway Press, 1996. 120~128.

[7] Demers, A., Keshav, S., Shenker, S. Analysis and simulation of a fair queue algorithm. Internetworking: Research and Experience, 1990,1(1):3~26.

[8] Zhang, L. Rate-Based scheduling discipline for packet switching networks. Electronics Letters, 1995,31(14):1130~1131.

[9] Karol, M.J., Hluchyj, M.G., Morgan, S.P. Input versus output queuing on a space division packet switch. IEEE Transactions on Communications, 1987,35(7):1347~1356.

[10] McKeown, N., Anantharam, V., Walrand, J. Achieving 100% throughput in an input-queued switch. IEEE Transactions on Communications, 1999,47(8):1260~1267.

[11] Anderson, T., Owicki, S., Saxe, J., *et al*. High speed switch scheduling for local area networks. ACM Transactions on Computer Systems, 1992,11(4):319~352.

[12] Simcoe, R., Pei, T. Perspectives on ATM switch architecture and the influence of traffic patterns on switch design. Computer Communication Review, 1995,25(2):93~105.

[13] Chiussi, F.M., Xia, Y., Kumar, V.P. Performance of shared-memory switches under multicast bursty traffic. IEEE Selected Areas in Communications, 1997,15(3):473~487.

[14] Zhang, H. Service disciplines for guaranteed performance service in packet-switching networks. Proceedings of the IEEE, 1995,83(10):1374~1399.

[15] Parekh, A.K,. Gallager, P.G. A generalized processor sharing approach to flow control in integrated services networks: the multiple node case. IEEE Transactions on Networking, 1994,2(2):137~150.

[16] Golestani, S. A self-clocked fair queueing scheme for broadband applications. In: Wieselthier, J.E., ed. Proceedings of the IEEE INFOCOM'94. New Jersey: IEEE Piscataway Press, 1994. 636~646.

[17] Stoica, I., Abdel-Wahab, H. An efficient packet service algorithm for high speed ATM switches. Journal of Computer Communications, 1998,21(9):839~852.

[18] Tabatabaee, V., Georgiadis, L., Tassiulas, L. QoS provisioning and tracking fluid policies in input queuing switches. In: Moshe, Sidi, ed. Proceedings of the INFOCOM'2000. New Jersey: IEEE Piscataway Press, 2000. 1624~1633.

Study on a Packet Scheduling Algorithm in High Performance Routers*

JIANG Yong, WU Jian-ping, XU Ming-wei

(*Institute of Network*, *Department of Computer Science and Technology*, *Tsinghua University*, *Beijing* 100084, *China*)

E-mail: jyong@csnet1.cs.tsinghua.edu.cn

http://netlab.cs.tsinghua.edu.cn

Abstract: Internet is facing two problems simultaneously: a faster switching/routing infrastructure and guaranteed quality-of-service (QoS). Each problem can be solved independently. High performance routers can be made faster by using input-queued crossbars instead of shared memory systems. QoS can be provided by using packet fair queuing (PFQ) algorithm. Until now, however, the two solutions have been mutually exclusive——all of the work on PFQ algorithm has required that routers use output-queuing or centralize shared memory. In this paper, on the basis of CIOQ (combined input output queuing) architecture, a packet scheduling algorithm DF^2Q (distributed feedback fair queuing) is designed and implemented. The most important feature of this algorithm is the introducing of feedback mechanism. the performance of DF^2Q is analyzed and discussed. Experimental results show that it can avoid internal congestion effectively and improve the efficiency of resource utilizing.

Key words: packet fair queueing; combined input and output queueing (CIOQ); feedback

* Received July 5, 2000; accepted October 16, 2000

Supported by the National Natural Science Foundation of China under Grant Nos.69682002, 69725003; the National High Technology Development 863 Program of China under Grant No.863-306-2D-07-01