一种支持多优先级的高速 Crossbar 调度算法

彭来献, 田 畅, 路 欣, 郑少仁
(解放军理工大学通信工程学院, 江苏南京 210007)

摘 要: 现有支持多优先级的高速 Crossbar 调度算法需要交互的控制信息较多, 控制信息的传输时间已成为调度算法性能提高的主要瓶颈. 为提高 Crossbar 调度的性能, 本文提出一种新的支持多优先级的高速 Crossbar 调度算法 p-iDRR, 该算法具有硬件实现简单、控制信息量少、高速和可扩展性强等优点. 仿真结果表明, p-iDRR 具有良好的吞吐量、时延性能, 适用于高速、多端口、大容量的路由器.

关键词: 高速路由器; 输入排队; 多优先级调度算法; p-iDRR; Crossbar
中图分类号: TP393.05 文献标识码: A 文章编号: 0372-2112 (2004) 08-1305-05

A New Scheduling Algorithm Supporting Multi-Priorities for High-Speed Crossbars

PENG Lai-xian, TIAN Chang, LU Xin, ZHENG Shao-ren
(*Institute of Communication Engineering, PLA University of Science and Technology, Nanjing, Jiangsu* 210007, *China*)

Abstract: Current scheduling algorithms with multi-priorities need to exchange a large amount of control messages whose transmission time has become the bottleneck for performance improvement of scheduling algorithms. To improve scheduling performance, we propose a new high-speed crossbar scheduling algorithm p-iDRR which supports multi-priorities. p-iDRR has many good features, such as being easy to implement, fewer control messages, high-speed and scalability. The results of simulation show that p-iDRR can achieve fine performance of throughput and delay and that p-iDRR is suitable to high-speed routers that have high-density ports and large capacity.

Key words: high-speed router; input-queueing; scheduling algorithm with multi-priorities; p-iDRR; Crossbar

1 引言

由于 Internet 规模和容量的飞速增长, 以及多业务的发展, 要求 IP 路由器不仅支持高速链路速率、具备大容量的处理能力, 还要能提供一定的服务质量(QoS)保证. 在路由器中, 交换网络负责查表后的数据的转发, 是制约路由器速度和容量的关键因素; 队列调度算法负责将到达的报文按照一定的顺序调度输出, 是路由器提供 QoS 保证的重要机制之一[1]. 传统的中、低速路由器大多采用输出排队的交换网络(包括共享缓存), 由于其可扩展性差, 不能适应高速的网络环境. 与此相比, 输入排队的 Crossbar 交换网络具有高速、良好的可扩展性等优点, 逐渐得到人们的青睐并被广泛应用于高速路由器[2~4]. 然而传统的具有 QoS 保证的队列调度算法(例如 WFQ, WF2Q 等[5])只适用于输出排队的交换网络, 不能直接用于输入排队的交换网络. 因此具有 QoS 保证的输入排队 Crossbar 调度算法已成为目前高速、大容量路由器研究中的一个热点.

目前, 支持 QoS 的输入排队 Crossbar 调度算法主要分为四类. 第一是基于带宽预约的调度算法, 为分组合理分配时隙, 并提供带宽或时延保证, 如 WPIM[6], BATCH-TSA[7] 等. 第二是提高 Crossbar 内部加速因子提供时延保证, 如 OCF[8], LOOFA-OCF[9] 等. 第三是完全模仿输出排队的调度, 从而提供与输出排队相同的 QoS 保证[10,11]. 第四是支持多优先级的调度算法, 如 p-iSLIP[12], OSP[13]. 上述算法普遍复杂度高、可扩展性差, 或者硬件难以实现, 限制了它们在高速、多端口、大容量路由器中的应用. 为了解决这些问题, 本文提出一种支持多优先级的调度算法——p-iDRR(prioritized iDRR), 该算法具有与 p-iSLIP、OSP 近似的性能, 但硬件实现相对简单; 另外, 算法收敛速度比 OSP 快, 克服了它们在实际应用中难以高速实现的缺陷.

2 问题描述

随着高速路由器技术的发展, 逐渐将交换单元(调度器和 Crossbar)与输入/输出接口卡分别放置于不同机架, 并使用光交换结构代替电子 Crossbar, 以支持几十甚至上百个高速接口[20]. 图 1(*a*)所示是一个 $N \times N$ 的输入排队 Crossbar 交换网

收稿日期: 2003-05-30; 修回日期: 2004-02-23
基金项目: 国家"863"计划资助课题(No. 2001AA121063)

络. 通常, 为了实现高速交换和控制上的方便, Crossbar 交换网络处理的数据单元为固定长度的信元, 一个信元通常取 64 Bytes 长度. 为了方便叙述和分析, 我们假设所有输入/输出端的速率相同, Crossbar 交换一个信元的时间间隔称为一个时隙.

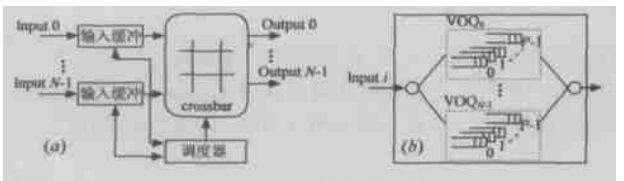

图 1 (*a*)输入排队 Crossbar 的交换网络结构;
(*b*)输入缓冲内的队列结构

调度器根据从输入端收集的控制信息, 每个时隙执行一次调度算法, 解决信元在输入/输出端的竞争, 避免信元的发送和接收冲突, 并控制 Crossbar 的开合, 建立输入与输出端的连接. 输入缓冲用于存储输入端到达的信元, 其内部的队列结构如图 1(*b*)所示. 为了消除队头阻塞(HoL: Head of Line Blocking), 队列结构采用了虚拟输出排队技术(VOQ: virtual output queueing), 即为每个输出端的每个优先级维护一个独立的 FIFO 队列. 其中优先级 $p\in\{0, 1, \cdots, p-1\}$, 0 为最高优先级, $p-1$ 为最低优先级. 因此, 每个输入缓冲内有 $N\cdot p$ 个 FIFO 队列. 信元到达过程是一个离散时间随机过程, 每个输入端每个时隙至多到达一个信元. 若在时隙 n 输入端 i 到达一个目的端为 j, 优先级为 p 的信元($0\leq i, j\leq N-1$, $0\leq p\leq P-1$), 那么该信元被放入 VOQ_j 的第$(p+1)$个子队列中 $Q_p(i, j)$中. 在本文中数据流均是单播数据.

支持多优先级的调度算法必须同时满足如下三个条件: (1)在一个时隙内每个输入端至多发送一个信元; (2)在一个时隙内每个输出端至多接收一个信元; 对于任一输入端 i 内任一 VOQ_j, (3)只有当所有子队列 $Q_m(i, j)$($0\leq m< p$, $p\leq P-1$)为空时, $Q_p(i, j)$才能被服务($0\leq i, j\leq N-1$).

3 p-iDRR 算法

p-iDRR 算法是对 DRR(Dual Round-Robin)[14]、iDRR(iterative DRR)[15] 算法的改进, 以支持多优先级调度. p-iDRR 解决输入/输出端竞争的方式、执行过程与 p-iSLIP 类似. p-iSLIP 每次执行过程包含多次迭代, 目的是为了提高时延性能, 每次迭代包括三个步骤(请求, 许可和接受)[12], 但 p-iDRR 每次迭代只包括"请求"和"许可"两个步骤.

在 p-iDRR 算法中, 每个输入/输出端都有一个仲裁器, 设有 P 个指针, 分别对应 P 个优先级. p-iDRR 也采用多次迭代的策略, 前面迭代中建立连接的输入/输出端不参加后面的迭代, 后面的迭代不影响前面迭代中已经建立的连接. p-iDRR 每次迭代包括两个步骤:

step1 请求. 输入端 i 的仲裁器有 P 个请求指针 r_{ip}($0\leq p\leq P-1$, $0\leq r_{ip}\leq N-1$), 分别指向当前优先选择的 VOQ. 如果输入端 i 中有发送请求, 首先找出所有具有最高优先级(假设为 p_i)的请求, 然后仲裁器使用指针 r_{ip_i}, 根据 round-robin 规则选择一个 VOQ 并向相应的输出端发送请求, 请求中携带优先级信息 p_i. r_{ip_i}仅在第一次迭代后更新. 如果该请求在 step 2 中被许可, 那么 r_{ip_i}等于被选择的VOQ 端口号加 1(mod N); 否则 r_{ip_i}等于被选择的 VOQ 端口号, 同时将该请求清除.

step2 许可. 输出端 j 的仲裁器有 P 个许可指针 g_{jp}($0\leq p\leq P-1$, $0\leq g_{ip}\leq N-1$), 分别指向当前优先选择的输入端. 如果输出端 j 接收到请求, 首先找出具有所有最高优先级(假设为 $p'_j=\min_i(p_i)$)的请求, 然后仲裁器使用指针 $g_{jp'_j}$, 根据 round-robin 规则选择一个输入端的请求, 并发送许可信息, 从而建立一个连接. $g_{jp'_j}$仅在第一次迭代后更新, 等于被选择的输入端口号加 1(mod N).

图 2 描述了一个支持 2 个优先级的 3×3 输入排队的 Crossbar 交换网络中 p-iDRR 算法一次迭代的过程.

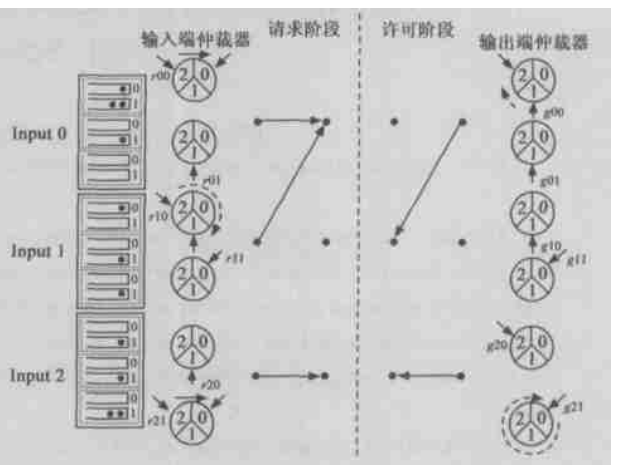

图 2 2个优先级的 3×3 输入排队的 Crossbar 交换网络中 p-iDRR 算法一次迭代的过程.

在请求阶段, 输入端 0和 1 同时向输出端 0 发送优先级为 0 的请求, 产生了输出端竞争. 输出端仲裁器使用 g_{00}许可了输入端 1 的请求, 我们注意到 g_{00}和 r_{10}分别更新为(1+1) mod 3= 2 和(0+1) mod 3=1; 而输入端 0 的请求未得到许可, r_{00}更新为 0, 同时这个请求会被清除. 如果再做一次迭代, 那么只有输入端 0 会向输出端 1 发出优先级为 1 的请求, 并得到许可, 最终建立连接. 由于不是第一次迭代, r_{11}和 g_{11}都不会更新.

图 3 表示了 p-iDRR 算法的实现框图. 每个输入端有一个输入端仲裁器, 图中右边虚线框内为调度器, 它主要由 N 个请求过滤器和 N 个输出端仲裁器组成. 请求过滤器的功能是

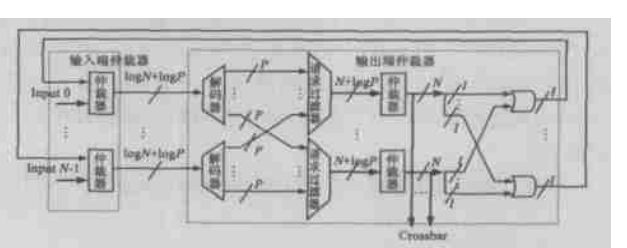

图 3 p-iDRR 算法的实现框图

对到达输出端的所有请求进行过滤, 找出所有最高优先级的请求, 并将这些请求和优先级信息送到输出端仲裁器. 输出端仲裁器根据优先级信息使用相应的指针, 从过滤后的请求中选择一个许可, 并将结果送到输入端.

3.1 复杂性分析

根据文献[12, 15]的分析, 我们规定 p-iDRR 一次执行过程也包含 log(N)次迭代. 下面分析 p-iDRR 算法一次迭代的时间复杂性. 在请求阶段, 每个输入端找出所有最高优先级的请求需要 $O(N\cdot P)$时间, 仲裁器从中选择一个 VOQ 需要 $O(N)$时间, N 个输入端总共需要 $O(N^2\cdot(P+1))$时间. 在许可阶段, 所有输出端至多接收到 N 个请求, 仲裁总共需要 $O(N\cdot(P+1))$时间. 因此, p-iDRR 算法一次迭代的复杂性为 $O(N^2\cdot(P+1))$.

3.2 与其它算法的比较

表 1 中比较了 p-iDRR、p-iSLIP 和 OSP 三种算法的性能. 其中控制信息量指在一次迭代中, 一个输入端与调度器之间交换的信息量. 复杂性指一次迭代的时间复杂性.

表 1　p-iDRR、p-iSLIP 和 OSP 三种算法性能的比较

算法	竞争解决方式		控制信息量(bits)			算法收敛需要的迭代次数	复杂性
	输入端	输出端	请求	许可	接受		
p-iSLIP	round-robin	round-robin	$N+N\cdot\log P$	N	$\log N$	$\log N$	$O(N^2\cdot(P+1))$
OSP	round-robin	串行轮询	$N+N\cdot\log P$	N	$\log N$	N	$O(N^2\cdot(P+1))$
p-iDRR	round-robin	round-robin	$\log N+\log P$	1	—	$\log N$	$O(N^2\cdot(P+1))$

考虑一个支持 8 个优先级的 64× 64 交换网络, 链路速率为 10Gbps. 一个 64 Bytes 长度的信元传输时间为 51.2ns, 也就是说, 调度器一次执行过程必须限定在 51.2ns 内完成. 根据表 1, 每次迭代, p-iSLIP 和 OSP 共有 326bit 的控制信息, 这些信息的传输将限制 p-iSLIP 和 OSP 的扩展并且可能降低其时延性能. 根据目前的技术, 串行电子电路很难达到 10Gpbs, 即使能够达到, 这些控制信息的传输也需要 32.6ns 的时间, 实际上允许调度器真正执行的时间只有 51.2- 32.6= 18.6ns. 这些时间无法满足下一次迭代的需要, 这将导致时延性能的下降. 另外, OSP 收敛需要 N 次迭代, 更加限制它在多端口路由器中的应用. 随着端口数目或优先级数目的增多, 留给调度器执行的时间更少. 与此相比, p-iDRR 算法只有 10bit 的控制信息, 显著地降低了控制信息量, 这些信息的传递不会成为调度器运行的瓶颈, 也不会牺牲信元的时延性能. 因此, p-iDRR 算法具有高速、可扩展性强等优点.

4 性能分析和仿真结果

由于在同样的参数设置下, 输出排队的交换网络具有最优的性能. 因此, 我们将输出排队的性能作为其它算法性能的参考. 为了绘图方便, 本文仿真模型采用一个支持 2 个优先级的 16× 16 的输入排队 Crossbar, 所有仿真的长度均为 100000 个时隙.

4.1 均匀分布业务流

均匀分布业务流是一种理想的业务流, 指每个输入端到达的信元均匀分布于各个输出端. 信元到达服从参数为 λ 的独立同分布的贝努里过程, λ 表示输入端流量负载, 即每个时隙内平均到达的信元数. 假设两个优先级比例为 1∶1. 在均匀分布业务流到达情况下, 图 4 比较了 p-iDRR 和 p-iSLIP(迭代次数都为 4)的平均时延性能. 从图中可以看出, 对于不同的优先级信元, p-iDRR 和 p-iSLIP 平均时延都非常接近. 最大吞吐量等于当交换机处于稳定状态临界点时负载的大小. 根据图 4, 交换网络的吞吐量由低优先级信元决定, p-iDRR 最大吞吐量在 97%左右, p-iSLIP 在 98%左右, 都接近 100%.

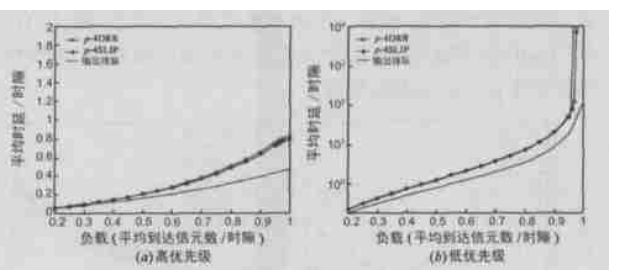

图 4　信元的平均时延性能比较

4.2 突发业务流

与均匀业务流相比, 突发业务流比较接近网络真实流量. 信元到达服务参数为 burst 的 ON/OFF 突发模型, burst 表示突发长度. 假设两个优先级比例为 1∶1. 图 5 比较了 p-iDRR 和 p-iSLIP 在不同突发长度的突发流到达情况下信元的平均时延性能. 在相同的业务流下, 两者的平均时延非常接近. 我们发现, 对于 p-iDRR 和 p-iSLIP, 高、低优先级信元的平均时延都与突发长度成正比. 这是由于当信元突发到达时, 队列中的信元得不到及时得调度, 造成等待时间的累积增加. 在实际网络环境中, 流量突发导致时延性能下降的现象普遍存在, 同其它调度算法一样, p-iDRR 也无法避免这一影响, 因此在使用 p-iDRR 时要使用流量整形等缓解突发的措施, 以提高系统的时延性能.

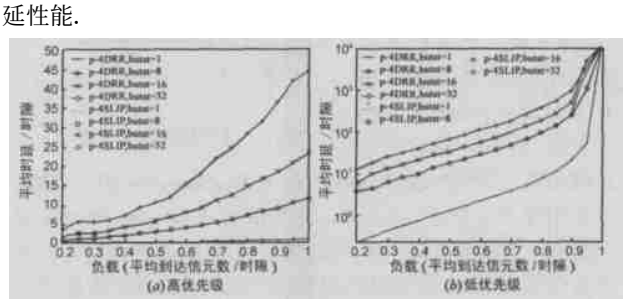

图 5　信元在不同突发流长度下的平均时延性能比较

4.3 不同比例的业务流

根据 p-iDRR 优先级调度规则, 有可能产生低优先级信元

“饿死”现象. 本节主要分析在不同比例的均匀业务流到达情况下, 信元平均时延性能的变化. 对于一个 M/G/1 多优先级队列假设优先级为 p 的信元到达速率为 λ_p, 服务速率为 1 个信元/时隙, 不同优先级的信元平均时延等于[16]:

$$\begin{cases} D_1=\dfrac{R}{1-\lambda_1} \\ D_2=\dfrac{R}{(1-\lambda_1)(1-\lambda_1-\lambda_2)} \\ \vdots \\ D_p=\dfrac{R}{(1-\lambda_1-\cdots-\lambda_{p-1})(1-\lambda_1-\cdots-\lambda_p)} \end{cases} \tag{1}$$

式中 R 表示平均剩余服务时间. 根据式(1), 若高优先级信元的数量越多, 则对低优先级的信元的时延影响越大. 我们仍假设有高、低两个优先级, 图 6 比较了它们在不同比例的均匀业务流到达情况下的平均时延性能. 从图中可以明显看出, 当高优先级信元数量较小时, 两者的平均时延也较小; 随着高优先级信元数量的增加, 两者的平均时延也增大, 性能下降. 因此, 使用 p-iDRR 算法时, 应尽量约束高优先级的流量, 提高交换网络的整体吞吐量和时延性能.

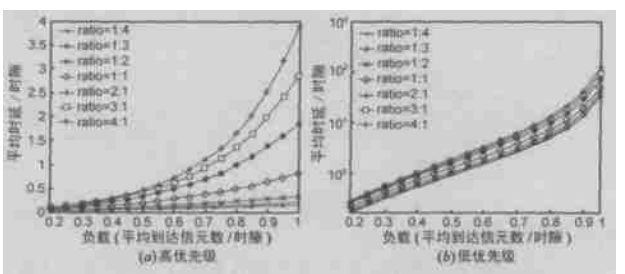

图 6 在不同比例的均匀业务流到达情况下信元的平均时延性能比较

5 仲裁器的实现

根据图 3 所示的 p-iDRR 实现框图, 请求过滤器和输出端仲裁器是调度器的核心部件. 由于过滤器是简单的组合逻辑电路, 其实现本文不再赘述, 本节主要介绍仲裁器的设计与实现. 输出端仲裁器的结构如图 7 所示, 主要由一个可编程优先级编码器(PPE: Programmable Priority Encoder)和一个指针寄存器组成, 指针寄存器中存有 P 个指针, 对应于 P 个优先级. *PPE* 是仲裁器设计的关键, 它的输入包括请求矢量 ***R*** 和 round-robin 指针. 仲裁器工作时, PPE 使用 round-robin 指针依据 round-robin 规则从 ***R*** 中选出一个请求进行许可, 将结果输出到 ***G***, f 表示是否有请求被许可. 除了输入/输出信号编码方式以及指针更新使能信号不同之外, 输入端仲裁器与输出端仲裁器的结构和设计基本一致. 我们设计并用 VHDL 语言实现了仲裁器, 选用了性能优良的温度计编码型 PPE[17,18]. 表 2 表示一个请求过滤器和一个 PPE 在不同规模下的面积耗费之和(等效逻辑门数目). *

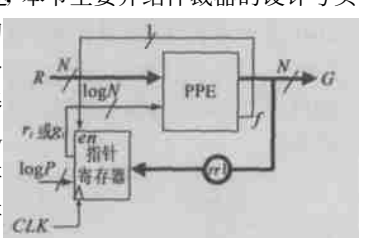

图 7 输出端仲裁器框图

表 2 一组请求过滤器和 PPE 的面积耗费

	$P=2$	$P=4$	$P=8$
$N=4$	126	264	480
$N=8$	306	528	816
$N=16$	834	1206	1752
$N=32$	1992	2610	3570

从表 2 中的数据可以看出, 面积耗费与端口数目和优先级数目成正比. 据此推算, 一个支持 8 个优先级 64×64 的调度器包含 64 组请求过滤器和 PPE, 面积耗费约为 50 万门. 调度器一次迭代的时间主要包含三个部分: 输入/输出端仲裁器、编码器和过滤器的逻辑延迟. 当前 0.1 um CMOS 电路的典型逻辑门延迟为 11.8ps, 一个 64×64 的 PPE 逻辑延迟大约为 2 ns 左右, 128×128 的 PPE 在 3ns 左右[19], 编码器和过滤器的电路简单, 延迟较小. 另外, 控制信息只有 10 bit, 其传输延迟也在 ns 级. 因此, 调度器一次迭代在 10 ns 以内, 如果采用流水线方式[17,18], 调度器将执行的更快, 可以支持 10 Gbps 以上的链路速率. 目前成熟的商用的 ASIC 芯片可以达到百万门级, 时钟频率 200 MHz 以上, 足以满足基于 p-iDRR 算法的调度器的需求. 如果减少迭代次数, 基于 p-iDRR 的调度器甚至可以支持具有 Tbps 交换容量的 128x128 交换网络. 此外, 请求过滤器和 PPE 都具有良好的可扩展性, 譬如两个 $M\times M$ 的过滤器(或 PPE), 可以通过增加简单的逻辑电路扩展为一个 $2M\times 2M$ 的过滤器(或 PPE), 限于篇幅, 具体扩展过程不再详述.

6 结束语

现有支持多优先级的输入排队 Crossbar 调度算法(如 p-iSLIP、OSP)需要大量的控制信息, 处理速度受限、实现复杂、扩展性差. 本文提出了一种支持多优先级的输入排队的 Crossbar 调度算法 p-iDRR. 与现有算法相比, p-iDRR 在保证了系统的吞吐量和时延性能的同时, 显著地减少了控制信息, 大大降低了实现复杂性, 提高了处理速度. 分析表明, p-iDRR 算法不仅具有良好的吞吐量、时延性能, 而且硬件实现简单, 高速、可扩展性强, 特别适用于高速、多端口、大容量的路由器.

参考文献:

[1] R Guerin, V Peris. Quality-of-service in packet networks: Basic mechanisms and directions[J]. Computer Networks, 1999, 31(3): 169—189.

[2] Nick Mckeown, et al. The Tiny Tera: A packet switch core[J]. IEEE Micro Magazine, 1997, 17: 26—33.

[3] Cisco Inc. Cisco 12000 series——Internet router product overview [Z]. http://www.cisco.com, 2001—10.

[4] Partridge C, et al. A 50-Gb/s IP router[J]. IEEE Trans. on Networking, 1998, 6: 237—248.

* 使用 Xilinx Foundation Series 3.1i 对 Xilinx SpartanII 系列 FPGA 综合的结果.

[5] H Zhang. Service disciplines for guaranteed performance service in packet-switching networks[J]. Proceedings of IEEE, 1995, 83(10): 1374—1396.

[6] D Stiliadis, A Varma. Providing bandwidth guarantees in an input-buffered crossbar switch[A]. Proc. Infocom'95[C]. Boston, MA, USA, 1995.3. 960—968.

[7] T Weller, B Hajek. Scheduling nonuniform traffic in a packet-switching system with small propagation delay[J]. IEEE/ACM Trans. Net, 1997, 5(6): 813—823.

[8] Anna Charny. Providing QoS guarantees in input buffered crossbar switches with speedup[D]. MIT, Sep 1998.

[9] Pattabhiraman Krishna, et al. On the speedup required for work-conserving crossbar switches[J]. IEEE Journal on Selected Areas in Communications, 1999, 17(6): 1057—1066.

[10] Shang-Tse Chuang, Ashish Goel, Nick McKeown, Balaji Prabhakar. Matching Output Queueing with a Combined Input Output Queued Switch[R]. Computer Systems Technical Report CSL-TR-98-758, March 1998.

[11] I Stoica, H Zhang. Exact emulation of an output queueing switch by a combined input output queueing switch[A]. Proceedings of IWQoS'98 [C]. Napa, CA, May 1998. 218—224.

[12] N Mckeown. Scheduling algorithm for input-queued cell switches[D]. UC Berkeley, May 1995.

[13] 孙志刚, 卢锡城. 一种用于区分服务路由器的 crossbar 调度算法[J]. 国防科技大学学报, 2000, 22(6): 52—56.

[14] H J Chao, J S Park. Centralized contention resolution schemes for a large-capacity optical ATM switch[A]. Proc. IEEE ATM Workshop [C]. Fairfax, VA, 1998.

[15] 彭来献, 田畅, 郑少仁. 高速 crossbar 控制算法 iDRR 及其性能分析[J]. 电子学报, 2003, 31(10): 1465—1468.

[16] D Bertsekas, R Gallager. Data Networks[M] second ed. Englewood Cliffs, NJ USA: Prentice Hall, 1992.

[17] 彭来献, 田畅, 郑少仁. 基于 iSLIP 算法的高速交叉矩阵调度器的 FPGA 设计与实现[A]. 第十五届南京地区研究生通信年会论文集[C]. 中国, 南京, 2000.475—480.

[18] Pankaj Gupta, Nick Mckeown. Design and implementation of a fast crossbar scheduler[J]. IEEE Micro Magazine, 1999, 19: 20—28.

[19] E S Shin, V Mooney, G F Riley. Round-robin arbiter design and generation[A]. Proceedings of the International Symposium on System Synthesis (ISSS'02)[C]. Kyoto, Japan, 2002.243—248.

[20] 彭来献, 李万林, 田畅, 郑少仁. 太比特路由器关键技术分析[J]. 电信科学, 2002, 18(3): 11—15.

作者简介:

彭来献　男, 1978 年 3 月出生于安徽阜阳, 1999 年于解放军通信工程学院获工学学士, 同年在该院以硕博连读的方式继续深造, 现在攻读通信与信息系统专业博士学位, 主要研究领域为高速路由器体系结构和高速交换网络.

田　畅　男, 1963 年 2 月出生于山东青岛, 博士, 副教授, 中国电子学会高级会员, 主要从事宽带交换技术、网络安全和无线分组网的研究, 近年在国内外有关刊物、会议上先后发表论文 40 余篇.