一种支持单播与组播混合业务的高速 Crossbar 调度算法

戴精科 彭来献 张邦宁
(解放军理工大学通信工程学院 南京 210007)

摘 要：当前在高速 crossbar 中支持单、组播混合业务调度的实用算法一般采用“请求-许可-接受”的处理流程(例如 ESLIP 算法)。研究发现，该类算法中存在单、组播“许可”相互阻塞现象，造成调度效率降低。从实用性出发，该文提出了一种新的支持单、组播混合业务的调度算法——ERGRR(Extended Request-Grant-based Round-Robin)，通过简化调度处理流程，克服了“许可”阻塞现象，提高了系统吞吐量、时延等性能。仿真结果表明，在单、组播混合业务流下，ERGRR 算法吞吐量、时延等性能优于 ESLIP 算法。另外，ERGRR 算法具有更好的公平性以及更加易于硬件实现。

关键词：路由器；输入排队；Crossbar；组播；ERGRR
中图分类号：TP393.05 **文献标识码**：A **文章编号**：1009-5896(2009)10-2299-06

A New Scheduling Algorithm Supporting Unicast and Multicast Traffic for High-speed Crossbars

Dai Jing-ke Peng Lai-xian Zhang Bang-ning
(*Institute of Communications Engineering, PLA University of Science and Technology, Nanjing* 210007, *China*)

Abstract: The current practical scheduling algorithms supporting unicast and multicast traffic in high-speed crossbars are generally based on a request-grant-accept process, such as ESLIP. But there is a phenomenon called “Grant” blocking between unicast and multicast cells in this kind of algorithms, which decreases the scheduling efficiency. According to the practicability, this paper presents a new algorithm supporting unicast and multicast traffic——ERGRR (Extended Request-Grant-based Round-Robin). ERGRR overcomes the “Grant” blocking and improves the system performance, such as throughput and delay, by simplifying execution process. The simulation results show that the ERGRR can achieve better performance of throughput and delay than ESLIP under various unicast and multicast traffics. In addition, ERGRR provides better fairness and its implementation complexity is lower than ESLIP.

Key words: Router; Input-queuing; Crossbar; Multicast; Extended Request-Grant-based Round-Robin(ERGRR)

1 引言

随着 Internet 组播业务(例如视频、音频会议)的迅速增长，人们迫切需要同时支持单、组播混合业务的高速路由器，其中交换结构是决定路由器速度、容量和性能的关键部件。目前高速路由器中一般都采用输入排队 crossbar 作为交换结构。通过控制 crossbar 交叉开关闭合，一个输入端能够同时与多个输出端建立连接，例如 ESLIP 算法[1]。因此，输入排队 crossbar 交换结构特别适用于单、组播业务高速处理环境中。

对于纯单播业务的输入排队 crossbar 调度，人们提出了众多性能优良、实用、硬件易实现的调度算法，例如 *i*SLIP[2]和 *i*RGRR[3]。相比而言，组播调度则更加复杂，文献[4]证明了最优组播调度是 NP 完全问题。文献[5]证明了即使采用虚拟输出排队(Virtual Output Queuing, VOQ)技术，组播调度算法也无法保证在各种业务流下均获得 100%的吞吐量。因此，输入排队 crossbar 组播调度一直以来是个难点问题，目前的研究重点主要从实用性考虑，寻找合适的排队机制以及性能良好的启发式算法。

鉴于上述原因，目前大多数文献中往往只考虑输入端维护 $k(1 \le k << 2^N - 1)$ 个组播队列[4,6−9]，并围绕提高吞吐量、时延、公平性等性能提出众多排队机制和调度算法。现有研究方法主要分为两大类：(1)单、组播使用相同的队列，统一进行调度[6,7]。文献[6]提出的 WBA 算法在每个输入端口只维护一个队列，吞吐量由于受到 HOL 阻塞而大大下降；为缓解 HOL 阻塞影响，文献[7]中采用多个队列 $(1 \le k \le N)$，虽然性能比 WBA 有明显改善，但是

2008-10-10 收到，2009-03-30 改回
江苏省自然科学基金(BK2007001)资助课题

必须解决多个队列下组播信元入队问题，该策略设计复杂，难以用硬件实现，实用性较差。(2)单、组播分别排队，分别进行调度[4,8,9]。这种方法采用一种隔离组播和单播业务的排队机制，即单播采用 VOQ 排队、组播采用单个 FIFO 排队，每个输入端只需要维护 $N+1$ 个队列。如何公平调度单、组播是这类问题关注的重点。文献[4]使用组播剩余的交换容量为单播服务，显然，这种方法对单播业务是不公平的；文献[8]中提出在每次调度时随机决定单播或组播优先服务，这种随机性难以适应两种业务流量的变化；为此文献[9]提出一种较为公平的调度策略，每次单、组播分别调度，通过一个“集成模块”将两个调度结果合并，被剔除的冲突的匹配边在下一次调度中默认选择，从而能够公平和最大限度利用交换带宽。然而“集成模块”需要保留上次调度结果，并且影响下一次调度结果，实现较复杂，仍然难以硬件实现。上述研究方法近期被应用于带缓存的 crossbar 组播调度中[10,11]，这里不再赘述。

上述大多数算法[6,8,9]均采用类似 iSLIP 算法的“请求-许可-接受”处理流程[2]，是一种寻找输入/输出端极大匹配的启发式调度策略，普遍存在排队机制或调度算法设计复杂，无法使用硬件实现，难以满足高速处理的要求，实用性较差。

至今得到成功应用的算法却是早在 1997 年提出的 ESLIP 算法[1]，被应用于 Cisco 12000 系列高速路由器中。ESLIP 算法是从 iSLIP 算法改进而来，采用多次迭代的“请求-许可-接受”处理流程，与大多数算法[6,8,9]相比，ESLIP 具有简单、高效和硬件易实现的优点。但本文分析发现在该算法中，如果某个输入端的单、组播请求同时得到了许可，那么必有一种业务未被接受，这会浪费大量的输出端许可，阻塞了输出端许可那些可能建立匹配的请求。我们称之为“许可”阻塞现象，这会造成调度效率降低，从而导致吞吐量、时延性能的下降。

为了消除“许可”阻塞现象，提高单播、组播混合业务调度性能，并从实用性角度出发，本文对 iRGRR 算法[3]进行改进，提出了一种新的高速调度算法 ERGRR(Extended Request-Grant-based Round-Robin)。该算法使用与 ESLIP 算法相同的排队机制。ERGRR 算法通过简化执行流程，每次迭代包含“请求-许可”两个步骤，在请求阶段只会发送一种业务的调度请求，从而避免了单播、组播业务之间的“许可”阻塞现象，提高了迭代过程中建立匹配的成功率。仿真研究结果表明，与 ESLIP 算法相比，当单播、组播业务同时到达时，在均匀、突发和非均匀等多种业务流下，ERGRR 算法能够获得更优的吞吐量、时延性能。另外，ERGRR 算法具有更好的公平性以及更加易于硬件实现，适用于高速、大容量、多端口的交换机/路由器(例如太比特交换机/路由器)。

2 相关背景

2.1 ESLIP 算法简介

ESLIP 算法考虑了多优先级调度，为简化分析，本文只考虑在一个优先级情况下算法的性能，在多个优先级的情况下，性能分析结果与一个优先级情况类似。ESLIP 算法一次执行过程包含 $\log(N)$ 次迭代，每次迭代按照“请求-许可-接受”3 个步骤进行：

步骤 1 请求(request)：如果一个未匹配的输入端有信元(不论是单播还是组播信元)等待发送，分别向对应的输出端发送所有的单播或组播请求。

步骤 2 许可(grant)：如果一个未匹配的输出端接收到多个请求，那么只保留单播或组播其中一种业务的请求。输出端仲裁器按照 round-robin 规则从中选择一个，并向对应的输入端口发送许可信号。(解决输出端竞争)

步骤 3 接受(accept)：如果输入端接收到多个许可，那么只保留单播或组播其中一种业务的许可。如果是单播许可，则输入端仲裁器按照 round-robin 规则从中选择一个；如果是组播许可，则一定接受。最后向发出该许可的输出端发送接受信号，这样就建立了一个匹配边。 (解决输入端竞争)

在“许可”步骤中，算法只能保留一种业务的请求进行许可，为了公平调度单、组播业务，ESLIP 算法采用单、组播在不同的时隙交替优先调度的策略。

2.2 “许可”阻塞现象

在 ESLIP 算法“请求”步骤中，输入端会发送所有的单、组播请求，这是“许可”阻塞现象出现的根本原因。这种现象在单播或组播优先调度的时隙内均存在，原理类似，这里通过实例分析单播优先调度时的情况。

图 1 展示了在一个 4×4 crossbar 中 ESLIP 算法存在的“许可”阻塞现象。图中实线、虚线分别表示单播、组播请求或许可。假设当前时隙优先调度单播业务，通用组播指针和所有单播指针均等于 0，指向输入端 0 或输出端 0。

从图 1(a)中可以看出，输入端 0 有一个到输出端 0 的单播请求，又有扇出为{1, 2, 3}的组播请求，输入端 1，2，3 分别有{0, 1}，{2, 3}，{3}的组播请求。此时各输出端优先许可单播请求，若没有单

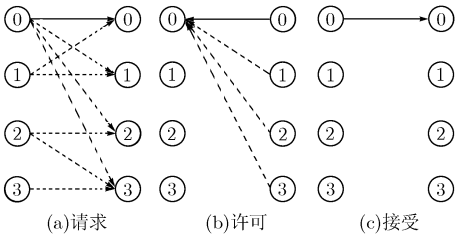

图 1 ESLIP 算法“许可”阻塞现象

播请求则许可组播请求，根据 round-robin 规则，输出端全部许可输入端 0，如图 1(b)所示。由于当前优先调度单播，输入端 0 只能接受其中的单播许可，致使输出端 1，2，3 发出的许可被浪费，同时阻塞了输入端 1，2，3 的请求被许可。由此可见，“许可”阻塞降低了一次迭代中建立匹配的成功率，使得调度效率下降，最终对吞吐量、时延等性能带来负面影响。

3　ERGRR 算法

3.1 算法描述

为了消除“许可”阻塞带来的调度效率下降问题，本文提出一种新的算法ERGRR，它是对iRGRR算法[3]的改进，以支持单、组播混合业务调度。ERGRR 算法一次执行过程包含多次迭代，每次迭代只有“请求-许可”两个步骤。同 ESLIP 算法一样，ERGRR 算法使用类似的 round-robin 仲裁方法和指针维护方式，也采用单、组播在不同的时隙交替优先调度的策略。ERGRR 算法的具体执行过程如下：

(1)在每次迭代开始时，将所有的输入和输出端都标记为未匹配。

(2)在每次迭代中：

步骤 1　请求：若一个未匹配的输入端有非空的 VOQ 等待发送信元，并且该 VOQ 对应的输出端空闲，则表示存在单播请求；若组播 FIFO 队列非空，并且队头信元的扇出对应的输出端有空闲，则表示存在组播请求。每个输入端仲裁器维护一个单播指针 r_i，r_i 指向当前优先选择的单播 VOQ。如果发送单播请求，则从 r_i 指向的位置开始，根据 round-robin 规则，仲裁器选择某个 VOQ 的请求并发送给相应的输出端仲裁器。r_i 当且仅当此请求在第一次迭代的步骤 2 中被许可才更新，等于当前被选择的 VOQ 端口号加 1(mod N)，否则不更新。如果发送组播请求，就把队头组播信元请求发送给相应的输出端仲裁器。 (解决输入端竞争)

步骤 2　许可：若一个未匹配的输出端收到单播和组播请求，则优先选择其中一种进行许可。每个输出端仲裁器维护一个单播指针 g_i，g_i 指向当前优先许可的输入端；所有输出端仲裁器共同维护一个通用组播指针 G_m，G_m 指向当前优先许可的输入端。如果许可单播请求，则从 g_i 指向的位置开始，根据 round-robin 规则，许可某个输入端请求，并发出一个许可信号，最后更新 g_i，等于当前被许可的输入端口号加 1(mod N)，否则不更新。如果许可组播请求，则从通用组播指针 G_m 指向的位置开始，根据 round-robin 规则，许可某个输入端请求，并发出一个许可信号。G_m 当且仅当此组播信元所有扇出请求在第一次迭代中被许可才更新，等于被选择的输入端口号加 1(mod N)，否则不更新。(解决输出端竞争)

(3)在每个时隙结束时，根据调度结果设置输入/输出端匹配标记，并配置 crossbar，建立输入/输出端连接，传送相应的信元。

特别提出的是，为避免“饿死”和指针同步现象[2]出现，单播指针和通用组播指针更新除满足上述条件外，还必须在对应业务优先调度时隙中进行。这里以 2.2 节所述的情况为例展示 ERGRR 算法一次迭代的执行过程，如图 2 所示。在第 1 次迭代中，输入端 0 存在单播和组播请求，由于当前时隙优先调度单播业务，因此在 “请求”步骤中，只发送了单播请求。而输入端 1，2，3 没有单播请求，则发送组播请求。在“许可”步骤中，输出端 0 优先许可单播请求，因此许可了输入端 0，并将单播指针 g_0 更新为 1；输出端 1，2，3 只收到组播请求，则许可之。由于此时 G_m=0，根据 round-robin 规则，输出端 3 许可了输入端 2 的组播请求。最终，输入端 0 得到了单播许可，将单播指针 r_0 更新为 1，其它单播指针不更新。虽然输入端 2 的组播扇出请求全部得到许可，但由于当前时隙不是组播优先调度，因此 G_m 也不能更新。

由于 ERGRR 算法在“请求”步骤中只发送一种业务的请求，因此避免了单、组播业务之间的“许可”阻塞想象。通过比较图 2(b)和图 1(c)所示的调度结果，可以发现 ERGRR 算法比 ESLIP 算法能够建立的更多的匹配边。根据下文第 4 节的性能分析，

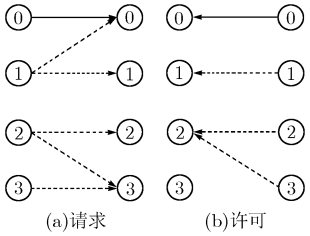

图 2 ERGRR 算法一次迭代的执行过程

再次验证 ERGRR 算法在单、组播混合业务下具有更好的性能。

3.2 与 ESLIP 算法的比较

根据文献[1-3]的分析，本文规定 ERGRR 算法一次执行过程也包含 $\log(N)$ 次迭代。表 1 中比较了 ERGRR 和 ESLIP 两种算法。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。

表 1 ERGRR 和 ESLIP 算法的比较

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
	输入端	输出端	请求	许可	接受	
ESLIP	round-robin	round-robin	$2N$	N	$\log(N)$	$\log(N)$
ERGRR	round-robin	round-robin	N	N	–	$\log(N)$

控制信息量多少是影响调度算法可扩展性的一个关键因素[3]。从表 1 中可以看出，与 ESLIP 算法相比，ERGRR 算法通过简化执行流程，减少了控制信息交互信息量。文献[3]指出在纯单播业务情况下，控制信息量越多，调度算法可扩展性虽然下降，但是可以获得更好的性能。然而在单、组播混合业务情况下，ESLIP 算法较多的控制信息却导致“许可”阻塞的出现，反而对时延、吞吐量等性能带来负面影响。因此，ERGRR 算法具有高效、可扩展性强等优点。

4 性能分析和仿真结果

对于输入排队 crossbar 单、组播调度算法的性能分析，由于解析分析的困难性，计算机仿真实验已成为交换结构和调度算法性能评价的重要手段[12]。本节使用计算机仿真的方法着重对 ERGRR 和 ESLIP 算法进行吞吐量、时延等性能的全面分析和比较。吞吐量是指在一个时隙内平均发送的信元数(×100%)，也等于输出端口平均利用率；时延是指信元在输入队列中平均等待时间，单位为时隙；负载 λ 表示输入端信元平均到达速率；对于组播信元，扇出(fanout) f_m 指该信元输出目的端口数目，平均扇出 $\overline{f}_m$ 是统计意义上所有组播信元的平均扇出[1]；组播比例 r_m 是指组播业务流量占总业务流量的比例。仿真采用 16×16 规模的 crossbar，算法一次执行过程包含 4 次迭代，组播信元平均扇出 $\overline{f}_m$ =4。仿真工具采用文献[13]提出的高速交换网络仿真系统，仿真长度均为 200,000 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。在均匀分布业务流到达情况下，图 3(a)比较了 ERGRR 和 ESLIP 算法在各种组播比例 r_m 下的平均时延特性。由图 3(a)可知，随着 r_m 的增大，输入端有效负载在增加，那么时延、吞吐量性能都会随之下降。在组播比例相同时，ERGRR 算法具有较小的平均时延，并获得更大的吞吐量。例如，当组播比例 r_m =0.1 时，ESLIP 算法的最大吞吐量为 75%，而 ERGRR 算法则可达到 80%。

4.2 突发业务流

与均匀分布业务流相比，突发业务流比较接近网络真实流量。信元到达服务参数为 bursty 的 ON/OFF 突发模型，bursty 表示突发长度。图 3(b)比较了 ERGRR 和 ESLIP 算法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，其中组播比例 r_m =0.03。由图 3(b)可知，在相同的业务流下，ERGRR 算法的平均时延小于 ESLIP，在高负载情况下，两者最大吞吐量都在 90%左右。对于 ERGRR 和 ESLIP 算法，信元平均时延都与突发长度成正比，这可以使用流量整形等缓解突发的措施来提高系统的时延性能。

4.3 非均匀业务流

为了研究 ERGRR 算法在更复杂业务流到达情

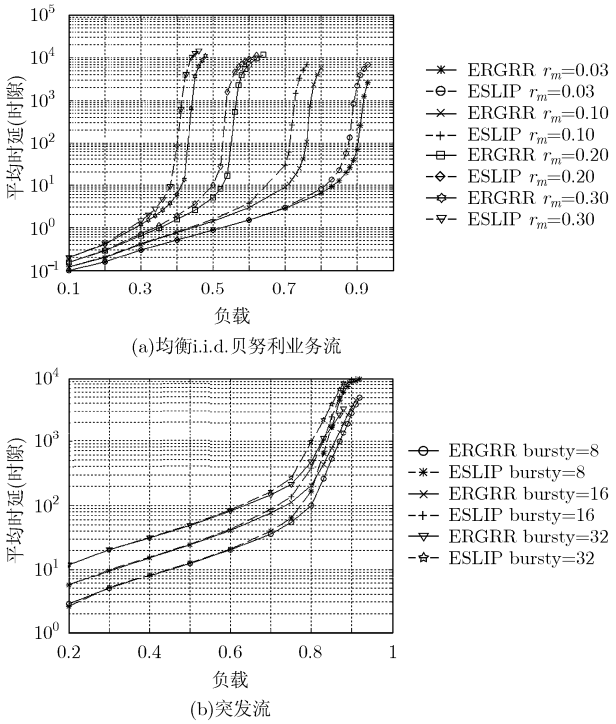

图 3 ERGRR 和 ESLIP 算法在均匀和突发业务流下的平均时延

况下的吞吐量，我们考虑如下通常采用的非均匀业务流模型。所有输入端负载相同，即 $\lambda_i = \lambda$，按下式分布于各个 VOQ：

$$\lambda_{ij} = \begin{cases} \lambda \cdot \left(w + \dfrac{1-w}{N} \right), & j = i \\ \lambda \cdot \dfrac{1-w}{N}, & j \neq i \end{cases}, \quad 0 \le i, j \le N-1$$

其中 $0 \le w \le 1$ 被称为非均匀(nonuniform)因子[3,10]。图 4 比较了 ERGRR 和 ESLIP 算法在不同非均匀因子 w 下的平均时延性能(组播比例 r_m =0.03)。由图 4 可知，在 w 较小时 ERGRR 算法具有较小的平均时延，但随着 w 的增加，二者的差别越来越小。这是因为在 w 较大时，到达某个输出端 j 的信元大部分来自于输入端 j，而其他的输入端的负载较轻，这样“许可”阻塞带来的影响不显著，特别是当 w 为 1 时，输入端之间不存在竞争，“许可”阻塞现象完全消失，而这两种算法的平均时延基本相同。在非均匀流量下，两者的吞吐量都有所下降，特别当 w =0.5 时，最大吞吐量只达到 75%左右。

4.4 公平性

一个信元从到达队头时刻开始，一直到被调度完毕时刻，其间的时间间隔被称为调度时延(单位：时隙)。crossbar 调度算法的公平性是指调度时延必须存在上限，否则算法就是不公平的。显然，调度时延越小说明算法的公平性越好。

文献[3]证明了 *i*SLIP 和 *i*RGRR 算法的调度时延上限分别为 $N^2 + (N-1)^2$ 和 N^2 个时隙。对于 ESLIP 算法，在最坏情况下单播和组播信元分别在各自优先的时隙内被调度。由于组播调度时全局“许可”指针只在某个组播信元被调度完毕(所有扇出完成调度)时才更新指针，那么一个队头组播信元最多只需等待 N 次调度即可完成交换。因此，ESLIP 算法在最坏情况下，单播信元调度时延上限是 $2(N^2 + (N-1)^2)$ 时隙，而组播信元为 $2N$ 时隙。综合上述，ESLIP 算法的调度时延上限为 $2(N^2 + (N-1)^2)$，同理可推出 ERGRR 算法的调度时延上界为 $2N^2$。因此，ERGRR 比 ESLIP 算法具有的更好的公平性。

5　结束语

从实用性角度出发，本文提出了一种支持单、组播混合业务的高速输入排队 crossbar 调度算法 ERGRR。目前支持单、组播混合业务的实用调度算法主要以 Cisco12000 系列路由器中使用的 ESLIP 算法为代表，一般采用“请求-许可-接受”的处理流程，存在单、组播业务“许可”阻塞现象，造成调度效率下降。ERGRR 算法通过简化处理流程，避免了“许可”阻塞现象发生，并且降低了输入端与调度器交互的控制信息量。仿真结果表明，在各种混和业务流下，ERGRR 算法具有比 ESLIP 算法更好的吞吐量、时延性能，并且提供更好的公平性和更加易于硬件实现。总之，ERGRR 算法具有良好的性能，按照现有的 ASIC 技术，基于 ERGRR 算法的调度器能够适用于高速、大容量、多端口的交换机/路由器。

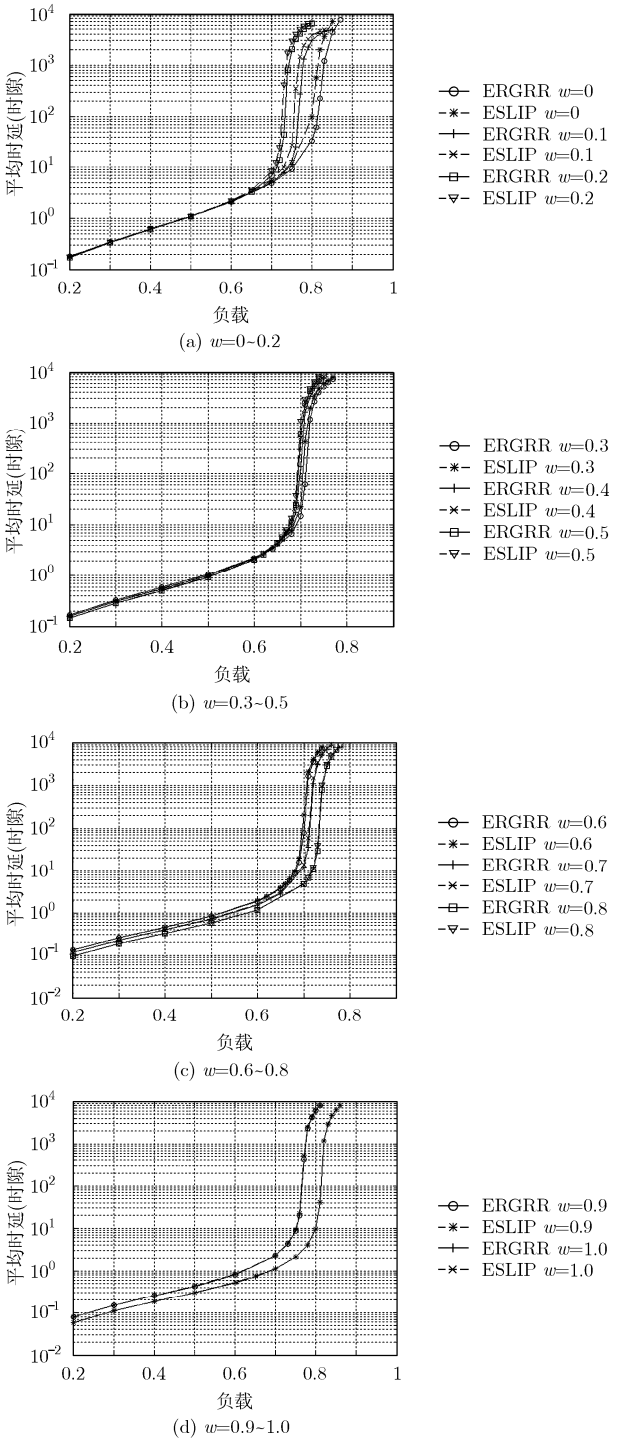

图 4 ERGRR 和 ESLIP 算法在非均匀业务流下信元的平均时延

参 考 文 献

[1] McKeown N. Fast switched backplane for a gigabit switched router[Z]. Cisco Systems white paper, http://www.cisco.com, 1997, 11.

[2] McKeown N. The *i*SLIP scheduling algorithm for input-queued switches[J]. *IEEE/ACM Transactions on Networking*, 1999, 7(2): 188–200.

[3] 彭来献, 田畅, 赵文栋. 一种具有 $O(\log N)$信息复杂度的高速 Crossbar 调度算法[J]. 电子学报, 2006, 34(11): 2024–2029.
Peng L X, Tian C, and Zhao W D. A new scheduling algorithm with $O(\log N)$ control messages complexity for high-speed crossbars. *Acta Electronic Sinica*, 2006, 34(11): 2024–2029.

[4] Andrews M, Khanna M, and Kumaran K. Integrated scheduling of unicast and multicast traffic in an input-queued switch[C]. Proceedings of IEEE Infocom'99, New York, USA, 1999, 3: 1144–1151.

[5] Marsan M A, Bianco A, and Giaccone P, *et al*.. Multicast traffic in input-queued switches: Optimal scheduling and maximum throughput[J]. *IEEE/ACM Transactions on Networking*, 2003, 11(3): 465–477.

[6] Prabhakar B, Ahuja R, and Mckeown N. Multicast scheduling for input-queued switches[J]. *IEEE Journal on Selected Areas in Communications*, 1997, 15(5): 855–866.

[7] 陈晴, 吴俊, 罗军舟. 一种适合于组播和单播的集成调度算法[J]. 计算机学报, 2004, 27(6): 758–764.
Chen Q, Wu J, and Luo J Z. An input-queued integrated scheduling algorithm for unicast and multicast traffic. *Chinese Journal of Computers*, 2004, 27(6): 758–764.

[8] Zhu W and Song Min. Integration of unicast and multicast scheduling in input-queued packet switches[J]. *Computer Networks*, 2006, 50(5): 667–687.

[9] Schiattarella E and Minkenberg C. Fair integrated scheduling of unicast and multicast traffic in an input-queued switch[C]. IEEE International Conference on Communications 2006 (ICC2006), Istanbul, Turkey, 2006, 1: 287–292.

[10] Mhamdi L, Gaydadjiev G N, and Vassiliadis S. Efficient multicast support in high-speed packet switches[J]. *Journal of Networks*, 2007, 2(3): 28–35.

[11] Giaccone P and Leonardi E. Asymptotic performance limits of switches with buffered crossbars supporting multicast traffic[J]. *IEEE Transactions on Information Theory*, 2008, 54(2): 595–607.

[12] 扈红超, 伊鹏, 郭云飞. 高性能交换与调度仿真平台的设计与实现[J]. 软件学报, 2008, 19(4): 1036–1050.
Hu H C, Yi P, and Guo Y F. Design and implementation of high performance simulation platform for switching and scheduling. *Journal of Software*, 2008, 19(4): 1036–1050.

[13] Stanford University. SIM manual[R]. http://klamath.stanford.edu/tools/SIM/, 2007, 10.

戴精科： 男，1984 年生，博士生，研究方向为卫星通信、卫星网络.

彭来献： 男，1978 年生，副教授，博士，研究方向为高速交换体系及其调度算法、Ad hoc 网络技术.

张邦宁： 男，1963 年生，教授，博士生导师，研究方向为卫星通信、信号处理、调制、编码和通信抗干扰.