Multistage-Based Switching Fabrics for Scalable Routers

Nian-Feng Tzeng, *Senior Member*, *IEEE*

Abstract—Rapidly growing demand for high-speed networks has prompted the investigation into scalable routers that are capable of forwarding data at the aggregate rate of multiterabits per second. Such a router contains many line cards (LCs) for admitting external links of various speeds. Those LCs are interconnected by a switching fabric to provide paths for packets to travel from arrival LCs to their respective departure LCs. The switching fabric employed in a router dictates the scalability and the overall performance of the router. It is thus crucial for future multiterabit routers to incorporate scalable switching fabrics capable of interconnecting large numbers of LCs. This work considers switching fabrics with distributed packet routing to achieve high scalability and low costs. Our fabrics are based on a multistage structure with different *recirculation designs*, where adjacent stages are interconnected according to the indirect *n*-cube connection style. They all compare favorably with an earlier multistage-based counterpart according to extensive simulation, in terms of performance measures of interest and hardware complexity. When queues are incorporated in the output ports of switching elements (SEs), the total number of stages required in our proposed fabrics to achieve a given performance level can be reduced substantially. The performance of those fabrics with output queues is evaluated under different "speedups" of the queues, where the speedup is the operating clock rate ratio of that at the SE core to that over external links. It is found via our simulation results that a small speedup of two is adequate for buffered switching fabrics comprising 4×8 SEs to deliver better performance than their nonbuffered counterparts with 50 percent more stages of SEs, when the fabric size is 256. The buffered switching fabrics under different traffic patterns are evaluated and discussed as well. Being scalable and of low costs, the proposed switching fabrics are ideally suitable for routers with large numbers of LCs.

Index Terms—Line cards, multistage interconnects, queue speedups, recirculation connections, routers, routing tags, scalability, switching fabrics.

1 Introduction

Routers serve to connect external links with various data rates through their line cards (LCs). On receiving a packet, an LC transmits it, based on the packet destination, to the outgoing LC through which the packet is to be delivered forward. The outgoing LC is determined by a table lookup, performed either locally at the arrival LC or via a remote forwarding engine (FE). It is thus basic for any router with multiple LCs (common in a current router [5], [31]) to employ a switching fabric to interconnect its constituent LCs, providing paths among LCs for packets to move from their arrival LCs toward their destined LCs. Switching fabrics naturally affect overall router performance and dictate router scalability.

Switching fabrics may be implemented on the basis of different approaches, including the bus, crossbar, shared memory, or multistage structure. A bus constitutes the simplest switch architecture, but it possesses limited scalability and speed since the shared single critical resource may easily become a bottleneck. For example, a backplane bus is utilized to link LCs and FEs of a Juniper M160 backbone router, for connecting up to eight LCs (which share four FEs). On the other hand, a crossbar fabric provides multiple simultaneous data paths over LCs, but it relies on a centralized scheduler to turn on and off cross-points to establish simultaneous paths for transmission. The cost and the cross-point control complexity of a crossbar grow rapidly as its size increases, limiting its scalability and speed. As an instance, Cisco 12000 Series routers [8] use crossbars (realized by hardware components resembling the PMC-Sierra TT1 chipset [23]) for interconnecting up to 15 LCs and one routing engine. Scheduling in such a crossbar follows a centralized algorithm similar to that proposed for the Tiny Tera [18].

A shared-memory switch configuration utilizes one storage for keeping incoming fixed-length packets, which are to be routed over their destined output ports. Several such switch designs have been reported in the ATM (asynchronous transfer mode) context, and they are based on linked lists, content addressable memory, or pipelined memory [10], [14], [24], [33]. A shared-memory-based fabric employs one shared storage, which limits aggregate forwarding rates up to hundreds of Gbps at most (even with pipelined memory design to achieve maximum bandwidth [14]). Obviously, such a fabric configuration is not suitable for scalable routers with the data forwarding rates possibly up to multiterabits per second.

As switching fabrics based on the bus or the crossbar do not scale, it is highly desirable to devise scalable fabrics for future high-performance routers which may contain large numbers of LCs. This is evidenced from the fact that the external link speeds of routers are expected to range from tens of Mbps to tens of Gbps (like OC-192 of 10 Gbps or OC-768 of 40 Gbps), and with such link speeds, a multiterabit router is

• *The author is with the Center for Advanced Computer Studies, University of Louisiana at Lafayette, P.O. Box 44330, Lafayette, LA 70504. E-mail: tzeng@cacs.louisiana.edu.*

Manuscript received 31 July 2002; revised 24 June 2003; accepted 10 Sept. 2003.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number 117048.

likely to involve a large number of LCs, even after aggregating slow links at each LC (e.g., the current Cisco 12000 routers permit each LC to terminate up to four links with an aggregate data rate capped at 10 Gbps).

A multistage switching structure is referred to as the space-division architecture and comprises multiple stages of switching elements (SEs) with or without buffers. Such a structure is self-routing and more cost-effective than its crossbar counterpart due to a lower hardware growth rate when the size increases, thereby promising better scalability. Different multistage-based configurations have been introduced for constructing scalable ATM switches, including various Banyan-type networks (like the Batcher Banyan [20], the Tandem Banyan [26], the Pipeline Banyan [32]) and the Shuffleout [1], and they may be adopted for router switching fabrics. Among them, the Shuffleout has been shown to be more cost-effective than Banyan-type networks [1]. A variation of the Shuffleout, formed by providing paths from its primary outputs back to its primary inputs, was considered. It is known as the closed-loop Shuffleout [2], which allows for packets to be recirculated back, when they fail to reach their destinations at the primary outputs. Both the Shuffleout and the closed-loop Shuffleout are nonbuffered. A nonbuffered multistage structure enjoys the advantage of hardware simplicity.

Buffers can be introduced to SEs in a multistage structure to enhance its throughput because they can hold packets which head for the same output port simultaneously and which otherwise have to be dropped or deflection routed [16]. Packets held in the buffers are delivered in sequence later on. For a multistage-based switching fabric, incorporating buffers in its constituent SEs can lower the number of stages required to achieve a given performance level. Buffers may be incorporated in the output (or input) ports of SEs, forming output (or input) queues. It has been shown that output queuing outperforms input queuing [13], but output queuing requires a "speedup," where the switching core of SEs runs faster than the connected links. Output queuing does not suffer from the *head-of-line* problem inherent to input queuing. Previous simulation and analytical studies indicated that a small speedup (say, 2-4) was enough in practice for output queuing to deliver packets quickly to their output ports [12].

In this article, we propose and evaluate switching fabric designs which not only are highly scalable, but also have lower hardware complexity than the Shuffleout, ideally suitable for scalable routers. Our proposed fabrics comprise multiple stages of SEs interconnected following the indirect n-cube connection style [21], with different approaches for *recirculating* packets from their primary outputs back into the fabrics. They make use of the I-Cubeout, which requires simpler SEs and fewer stages to achieve a given packet drop rate than a corresponding Shuffleout [28]. The simplest recirculation design requires little control logics for recirculating paths, but exhibits relatively lower performance. The other two recirculating approaches make use of more control logics to arrive at better performance. Our simulation has unveiled that the proposed switching fabrics (resulting from three recirculating approaches) all outperform their compatible closed-loop Shuffleout, in terms of 1) packet mean latency for any given load and 2) the packet drop rate for a given number of stages. It is also demonstrated by the simulation study that a single fault leads to negligible performance degradation. Note that the recirculating approaches proposed here are equally applicable to the (open-loop) Shuffleout [1] and other multistage-based structures for reducing the number of stages required to achieve a specified performance level. When applied to the Shuffleout, however, those approaches yield designs expected to outperform their close-loop Shuffleout counterpart, due to recirculating packets over the later stages (which carry less traffic than the earlier stages).

When buffers are introduced to output ports of SEs, it is found that the number of stages for our proposed fabrics to yield a given performance level is reduced substantially, even under the speedup of only two. The proposed switching fabrics with buffers are shown to be capable of handling various traffic patterns likely to exist in typical routers. Since the costs of our switching fabrics are basically proportional to the number of stages involved, buffered fabrics appear more attractive than their nonbuffered counterparts and are better applicable to large sized routers with hundreds of LCs. Note that buffered switching elements require an appropriate scheduling mechanism to be incorporated for handling packets on the basis of per flow or per class to ensure differentiated quality-of-service (QoS) guarantees.

Considerable research has been devoted to providing QoS support in the network for various applications [3], [25], addressing such issues as traffic arrival policing and shaping, packet dropping schemes, traffic classification and regulation, and buffer management and scheduling. Our proposed switching fabrics with appropriate scheduling and dropping at each SE with buffers would help to realize per-hop service guarantees since such additional scheduling and dropping at the SE level is performed in addition to conventional managing and scheduling at the switching fabric level for those buffers placed at the primary inputs and/or primary outputs (like that described in [6], as an example). These per-hop guarantees at each individual router naturally facilitate end-to-end service guarantees in the network. While more sophisticated management and scheduling techniques may be required for buffers at the fabric level (e.g., weighted fair queuing, earliest-deadline-due scheduling), a simple class-based queuing is believed to be adequate for the buffers in each SE, with QoS scheduling carried out perhaps once over a certain number of packets (defined by the sampling interval, rather than every packet) to avoid slowing down the SE clock rate and to keep SE hardware complexity acceptably low. While our switching fabrics can benefit and enhance QoS support, their QoS aspects will not be explored in this article.

It should be noted that one may choose to limit the size of switching fabrics for terabit routers to 32, by following the crossbar configuration with centralized cell scheduling, like the Yuni switch architecture [34] or the switch fabric design based on CMOS transceivers [29]. This way scales up router performance via increasing the degree of aggregation at the LCs, pushing up the crossbar core speeds accordingly. For example, the Yuni switch resorts to the LC speed of 80 Gbps, in contrast to the 10 Gbps adopted by Cisco 12000 Series routers and the Juniper M160 router. Such a high-speed LC is expensive to manufacture, and hardware complexity involved for this degree of link aggregation can be very high, limiting its further growth in the data forwarding rate and, thus, router scalability.

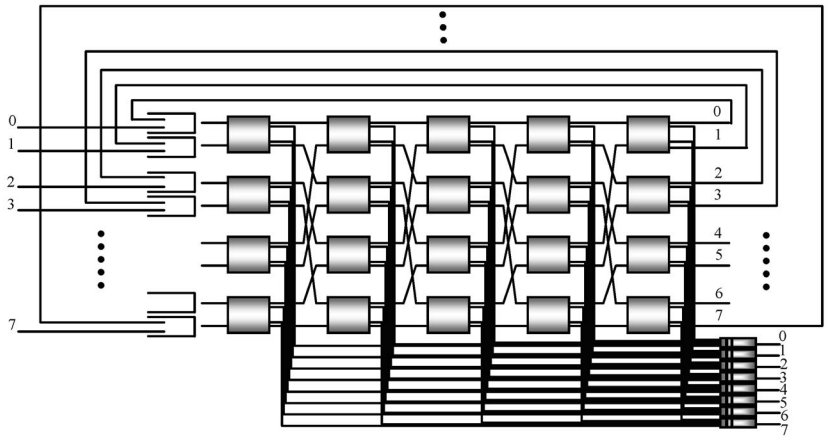

Fig. 1. A closed-loop shuffleout of size 8 with five stages of SEs (from [2]).

2 Review of Shuffleout and Its Variation

This section first briefly reviews the Shuffleout [1] and its variation (i.e., the closed-loop shuffleout [2]), which are implemented on the basis of a multistage structure without buffers. The Shuffleout switch consists of multiple stages of nonbuffered SEs, which are basically crossbar switches, with adjacent stages interconnected by a shuffle connection pattern. An SE has switch outlets terminating at its output queues, which are fed by packets (of fixed length) received on links of stages where they reach their destined rows. If a packet fails to reach its destined row after traversing all the stages, the packet is dropped at the output side of the last stage. A Shuffleout switch of size $(N =)8$ composed of 2×4 SEs involves $3(= \log_2 N)$ or more stages of SEs, where each SE has two switch outlets terminating at two output queues and each queue is fed by packets received at those constituent stages. There are four SEs in each stage of such a Shuffleout switch.

In order to perform routing over a shortest path for a packet in the Shuffleout switch, each packet is provided with an additional field, called *tag*, which specifies the address of the required switch outlet for the packet. On receiving a packet, the SE always attempts to route the packet along the shortest path to the destined switch outlet, where the shortest path involves the fewest interstage hops needed to reach its destined outlet. If two packets at an SE require the same switch outlet to an SE in the next stage (or to an output queue), called a *remote* switch outlet (or a *local* switch outlet), deflection routing is applied so that the packet closest to its destined switch outlet is routed along the shortest path, whereas the other packet is deflection-routed over another remote switch outlet. When a packet arrives at an SE in any stage of the Shuffleout switch, its tag field is updated by the distance computing block of the SE through recomputing the distance of the packet from the current SE to its destined SE for identifying the SE remote switch outlet which is on the shortest path [9]. This recomputation of the tag field is necessary at every SE, whether the packet is routed over a shortest path or by deflection routing.

The closed-loop Shuffleout [2] was formed by adding recirculation paths to the open-loop structure so that packets may travel through all the stages several times until they reach their destined output queues eventually (or get dropped when storage at each primary input for holding reentered packets is fully exhausted), where a *primary input* is an input of the first stage. Storage is provided at each primary input for holding packets which arrive concurrently, but there is no queue at the input or output ports of its constituent SEs. A closed-loop Shuffleout of size 8 with five stages of SEs is shown in Fig. 1, where a buffer is incorporated in each primary input to hold concurrent arrivals [2]. Like the Shuffleout, the closed-loop Shuffleout achieves self-routing in each SE by making use of the whole tag (known as address-based [2], as opposed to bit-based for the Banyan-type networks and our switching fabrics based on the I-Cubeout [28]), meaning that the entire tag, not a single bit, must be processed in order to route a packet. Every SE is equipped with the distance computing blocks, one for an input port, to achieve address-based self-routing. A diagram of such a block is given in [9]. The complexity of the block grows in the order of approximately $O(m^3)$, where m equals $\log_2 N$ for a switch of size N comprising 2×4 SEs. This rapid growth in hardware complexity makes it prohibitively expensive to construct SEs for a large (closed-loop) Shuffleout switch, limiting scalability. In addition, the commutation block [9] in every SE of the (closed-loop) Shuffleout becomes more complicated when the size of SEs increases, since it is fed with outputs of all distance computing blocks in an SE.

Unlike a Banyan-type network (of size, say N), where multiple copies of the Banyan (each with $\log_2 N$ stages) are cascaded horizontally, stacked vertically, or preceded by a complicated sorting and/or a random network, the Shuffleout permits any number of stages of SEs, not necessary to be an exact multiple of $\log_2 N$. Distance computation in each SE of the Shuffleout enables the packets to reach their destinations sooner than a random mechanism in the event of conflicts, making the Shuffleout take fewer stages to attain a given packet drop rate than its Banyan-type counterparts. On the other hand, the I-Cubeout structure is demonstrated to require slightly fewer stages to exhibit a given drop rate than it comparable Shuffleout [28], despite it is bit-based self-routing without distance computation with simpler routing logic and better scalability, as detailed subsequently.

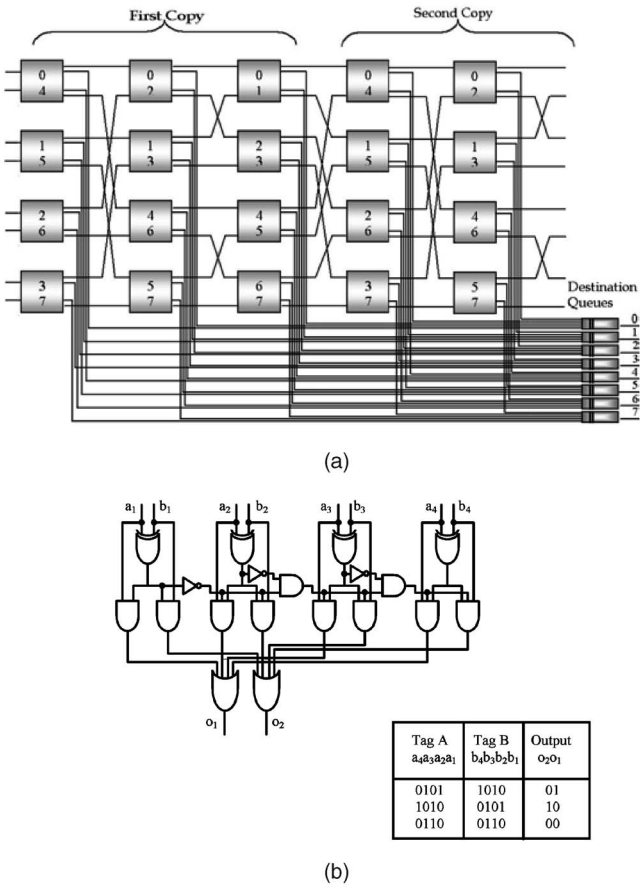

Fig. 2. (a) I-Cubeout of size 8 (ICO_8) with five stages and (b) 4-bit priority resolution logic.

3 Proposed Switching Fabrics

Our proposed switching fabrics for scalable routers are based on the *I-Cubeout* [28] (or ICO for short), which comprises multiple stages of $b \times 2b$ SEs, with b outlets of each SE being remote (for connecting SEs in the next stage) and the other b outlets being local (for terminating packets at output queues). For simplicity, we explain in the following, the case of $b = 2$ only.

3.1 The I-Cubeout

An I-Cubeout switch with size 8 (denoted by ICO_8) is depicted in Fig. 2a, where each SE is provided with outlets for terminating packets at their destination queues as soon as they reach their destined rows [28]. Each stage contains four SEs, which are small crossbar switches, each with two remote outlets (for connecting SEs in the next stage) and two local outlets (for terminating packets at destination queues). Adjacent stages are interconnected according to an indirect n-cube connecting pattern [21]. Specifically, the two remote outlets of any SE in the same stage differ in their addresses by a constant; those in stage 1 (i.e., the leftmost stage) differ by $N/2$, those in stage 2 differ by $N/4$, and so on, for ICO_N ($N = 2^n$). At the nth stage, the two remote outlets of any SE differ by 1. A *copy* of the indirect n-cube connection spans from stage 1 to stage n, as indicated in Fig. 2a. In stage n+1, the two remote outlets of an SE differ by $N/2$, identical to the situation of stage 1. This stage begins another copy of the indirect n-cube connection, which covers the next n stages. ICO_N may contain any number of stages, not necessarily an integer multiple of n. For example, ICO_8 may contain five (5) stages as depicted in Fig. 2a, while a copy of the indirect 3-cube covers three (3) stages.

Packets in ICO are self-routed in a distributed manner, with the routing tag of each packet computed at the primary input according to its source and destination addresses via a bit-wise XOR operation. The tag represents the positions needed to be "corrected" before reaching its destination and is carried along with the packet to guide its traversal across ICO. The first tag bit is used by SEs in stage 1 of any copy, and the second tag bit is by SEs in stage 2 of any copy, etc. If a tag bit, say in position p, is "1," the packet takes a "cross" state at the visited SE in stage p of any copy, signifying that the upper (or lower) inlet of the SE is connected to the lower (or upper) remote outlet. It reflects that the packet is "corrected" at position p and the correction can be done in stage p of any copy. After it is done, the pth tag bit is reset to "0." If the tag bit is "0," the visited SE is set to the "straight" state since no correction is then needed. After a packet advances one stage, its tag is rotated leftward by one bit position, so that the tag bit to be examined at any stage is always the leftmost one. When the tag bits all equal "0," the packet has reached its destined row and may take the associated local outlet to reach its destination queue.

Consider ICO where no buffers are equipped at SE outlets and every outlet can accept only one packet in each cycle. If two packets at an SE in stage p have distinct values in their pth tag bits, a conflict occurs and only one of the two conflicting packets is forwarded to its destined outlet, with the other being deflection-routed. The packet with a smaller distance to its destination is given priority, where the *distance* is defined as the minimum number of stages a packet has to travel before getting to its destination. The distance of a packet can be told directly from its current tag. Specifically, the distance of a packet is the rightmost nonzero bit position of its tag (given that the tag has been rotated leftward by r bit positions if the packet is at stage (r+1) of any copy). The rationale behind giving priority to the packet with a smaller distance is to let that packet reach its destination as soon as possible, and this is more likely to achieve because it has fewer stages left to travel, freeing resources for other packets. If two conflicting packets have an identical distance to their respective destinations, a random one is chosen to give priority, with the other one deflection-routed. If a deflection-routed packet conflicts with a nondeflection-routed one at an SE in a later stage, the former always has a larger distance and, thus, lower priority, than the latter. This naturally keeps deflection-routed packets as few as possible, making best utilization of resources.

Unlike the Shuffleout, ICO never recomputes the tag of a packet, even after the packet is deflection-routed at an SE. This makes the constituent SEs of ICO much simpler and more scalable than those of the Shuffleout. Every SE in the Shuffleout switch is equipped with the distance computing blocks, one for an input port. The complexity of such a computing block grows in the order of approximately $O(m^3)$, where m equals $\log_2 N$ for a switch of size N comprising 2×4 SEs [9]. This rapid growth in hardware complexity makes it prohibitively expensive to construct SEs for a large Shuffleout switch, limiting scalability. The priority resolution logic in each SE of ICO is realized by comparing the distances of two

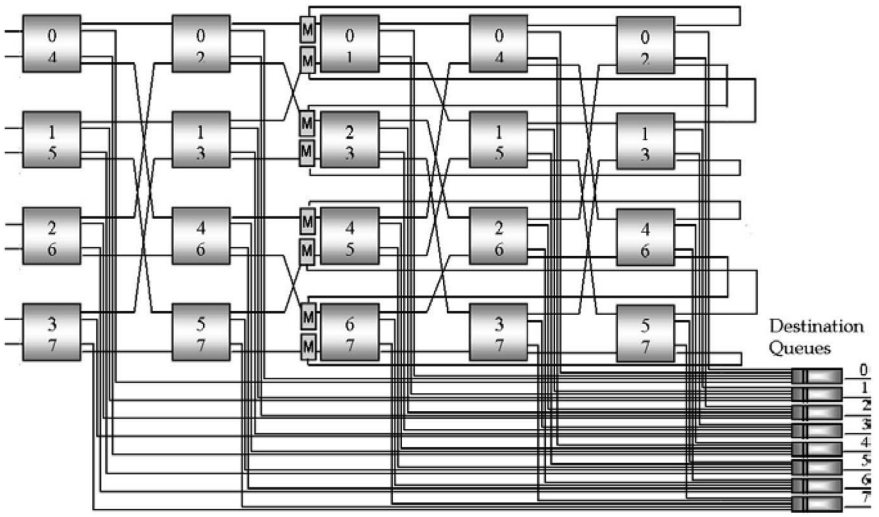

Fig. 3. ICO_8 with static recirculating connections, $\mathrm{ICO}_8^{\mathrm{S}}$.

tags in an SE, as illustrated in Fig. 2b, where output $o_2o_1 = 01$ (or 10) indicates that tag $A = a_4a_3a_2a_1$ has a larger (or smaller) distance than tag $B = b_4b_3b_2b_1$, and output 00 indicates identical tags. ICO not only exhibits lower hardware complexity than its (open-loop) Shuffleout counterpart (mainly because every SE in the Shuffleout switch is equipped with *b distance computing blocks*, one for each input), but also compares favorably with a previous switch of size 16 fabricated on a single chip [4], [30], in terms of the transistor count and power consumption [27].

It should be noted that deflection-routing in a 2×4 SE takes place as stated above, if no buffers are incorporated therein. For SE with output queues and with a speedup of k (≤ 2), up to k competing packets are accepted by any output queue and deflection-routing happens only if there are more than k packets competing for one output port or if the targeted output queue has a capacity less than what is needed to hold k packets. While ICO explained so far is composed of 2×4 SEs, it is obvious that ICO in general can be built by $b \times 2b$ SEs, each connecting to b SEs in the next stage and terminating at b destination queues. When a packet has not reached its destined row at an SE, it is forwarded to the next stage through the remote outlet decided by the leftmost $\log_2 b$ routing tag bits. If a packet fails to reach its destined row after traversing all the stages of ICO, the packet is either dropped at the output side of the last stage (called the *primary output*s) or forwarded back into ICO if recirculating connections exist from ICO primary outputs back to ICO. In the following, ICO with appropriate recirculating connections are denoted as our proposed switching fabrics.

3.2 Packet Recirculation Approaches

The recirculating connections of ICO enable the packets to reenter the fabrics when they fail to reach their destined rows at primary outputs, instead of being dropped otherwise. They aim to lower the number of stages required in ICO for a given performance level, when compared with ICO without such recirculating connections, in order to simplify the destination queues logics and to reduce the total number of stages (of SEs). For efficient utilization of fabric resources, the recirculating connections in ICO are to be fed to the *last* copy of fabrics either statically or dynamically, unlike the closed-loop Shuffleout which recirculates connections back to the fabric primary input side statically. Our ICOs all lead to fewer conflicts and, thus, less resource wastage when using the last copy of stages for routing reentered packets to their destined rows. This is because traffic is consistently lighter in a later stage of ICO_N, and one full copy (of $\log_b N$ stages) is needed to route any packet to its destination. From this point onward, a packet is designated as a *fixed-length data unit* which can be delivered from one SE to another SE in a subsequent stage in one cycle. (In some papers, such a data unit is referred to as a cell instead, but the term of cell is coined specifically for the ATM data unit and is thus not adopted here to avoid confusion.) Messages are of varying sizes and are fragmented into packets (of fixed length) before being injected into switching fabrics for delivery. At their destinations, packets are put back to their original messages before being processed at the outgoing line cards (e.g., producing appropriate headers and performing needed fragmentation according to the protocols employed therein). Different recirculation designs for ICOs are described subsequently.

3.2.1 Static Recirculating Connections

The simplest form of recirculation is realized by connecting each primary output (at the last stage) back to the input of $\log_b N$ stages ahead in the same row, for ICO_N composed of $b \times 2b$ SEs. It has a *static* reentry point for each recirculating connection, denoted as $\mathrm{ICO}_N^{\mathrm{S}}$, irrespective of whether or not there is a packet arriving from a prior stage at that reentry point. $\mathrm{ICO}_8^{\mathrm{S}}$ with five stages of 2×4 SEs is illustrated in Fig. 3, where the reentry point of each recirculating connection is three stages ahead (of the primary output side). Making recirculating connections this way involves the least extra logics. Each reentry point is equipped with a 2-to-1 multiplexor, denoted by "M" in the figure, to accommodate the recirculating connection.

The routing tag of each packet sent along a recirculating connection back to the switching fabric *need not* be recomputed or modified, as the recirculated packet is treated as if it advances to a next stage. There is no buffer provided at each reentry point, and the packet can be fed through a recirculating connection back to $\mathrm{ICO}^{\mathrm{S}}$ only if there is *no packet arriving* from the prior stage to that reentry point concurrently. This means a reentered packet has lower priority, and resources

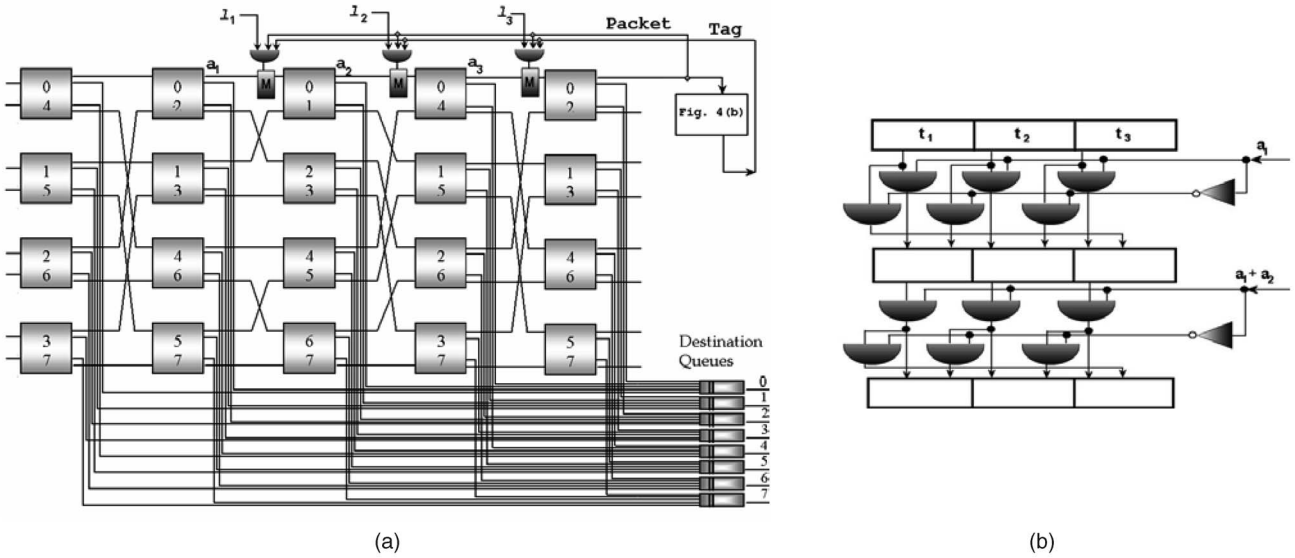

Fig. 4. (a) ICO_8 with dynamic recirculating connections, $\mathrm{ICO}_8^{\mathrm{FA}}$ and (b) tag rotation logic for a recirculating connection.

are made available to reentered packets only if they are otherwise not utilized. Note that, if recirculated packets are fed back to the first stage of fabrics, like the closed-loop Shuffleout, all such packets are dropped when the load equals 1.0 where a new packet is always injected into every primary input in each cycle, causing the recirculated packets (which are older) to be discarded. After a packet gets reentered into the fabric but conflicts with another packet later at any SE (in the last copy of $\log_b N$ stages), the one closer to its destination is given priority, irrespective of its age. If their distances are identical, the older packet (i.e., the reentered one) gets priority. This is so desired to keep the *worst packet latency* small in ICO with recirculating connections and is the only extra logic added to SEs when recirculating connections exist.

3.2.2 Dynamic Recirculating Connections—First Avail

In order to lower the possibility of dropping reentered packets due to conflicting with other concurrent packets from the prior stage, every recirculating connection is provided with $\log_b N$ entry points, one for each stage of the last copy, as shown in Fig. 4a. Each entry point is equipped with a 2-to-1 multiplexor to accommodate a recirculating connection. When a packet is to reenter the fabric, it utilizes the entry point which corresponds to the *first* available inlet, where an inlet is available (at the time when a packet is to reenter the fabric) if the prior stage delivers no packet to the inlet simultaneously. As the outlet of each SE has a latch to hold one packet at the end of each cycle, the availability of an inlet can be told directly using the latch indicator of the outlet (in the prior stage) which is connected to the inlet. If none of the inlets in the last copy is available, the packet is dropped. This fabric design recirculates packets back via the first available entry points dynamically, referred to as $\mathrm{ICO}^{\mathrm{FA}}$. Such a design is expected to get more packets recirculated successfully back to the fabric at the expense of more multiplexors and their control logics.

Consider a given recirculating connection. Let the availability of an inlet at stage i of the last copy be denoted by a_i (with $\mathrm{a}_i = 1$ signifying that the inlet is available), the control signal for loading the packet via the entry point at stage i along the recirculating connection shown in Fig. 4a is expressed as $l_i = \underline{\mathrm{a}}_1\ \underline{\mathrm{a}}_2\ \ldots\ \underline{\mathrm{a}}_{i-1}\ \mathrm{a}_i$, where $\underline{\mathrm{a}}_j$ represents -a_j for $1 \le j \le i-1$. Note that Fig. 4a provides only the top recirculating connection for clarity, leaving out the remaining connections. These control signals ensure that a recirculated packet reenters the fabric through the first available entry point (and only that one). When a packet is fed back at stage f, its tag has to be rotated leftward by $(f-1) \times \log_2 b$ bit positions before reentry. A fast and simple logic to achieve such tag rotation is depicted in Fig. 4b. The logic involves $\log_b N$ registers of length $\log_2 N$, with an appropriate control signal for data paths between a pair of adjacent registers, as given in the figure. After the packet reenters the fabric with its proper tag accompanied, it is routed following the same routing algorithm. If it conflicts with another packet in an SE, the associated priority resolution logic is invoked to decide the one to be routed to its desired outlet, with the other being deflection-routed.

In general, $\mathrm{ICO}^{\mathrm{FA}}$ does not have to provide entry points for *all* stages in the last copy, but just the early few stages, to restrict hardware overhead with almost no impact on performance measures of interest. It is found by our simulation that providing three (a constant number of) entry points is almost as good as providing $\log_b N$ entry points for any $\mathrm{ICO}_N^{\mathrm{FA}}$, presenting good scalability. We thus refer to $\mathrm{ICO}_N^{\mathrm{FA}}$ as the design with *three entry points* per recirculating connection (which translates to constant hardware overhead) subsequently. Note that $\mathrm{ICO}_N^{\mathrm{S}}$ may be viewed as a special case of $\mathrm{ICO}_N^{\mathrm{FA}}$, where only one entry point is provided for each recirculating connection; in this case, no tag rotation is necessary.

The additional two entry points along each recirculating connection in $\mathrm{ICO}_N^{\mathrm{FA}}$ enhance throughput (or, equivalently, offered load) noticeably, in comparison to that of $\mathrm{ICO}_N^{\mathrm{S}}$. In this article, the terms of throughput and offered load are used interchangeably. The throughput gain is more pronounced

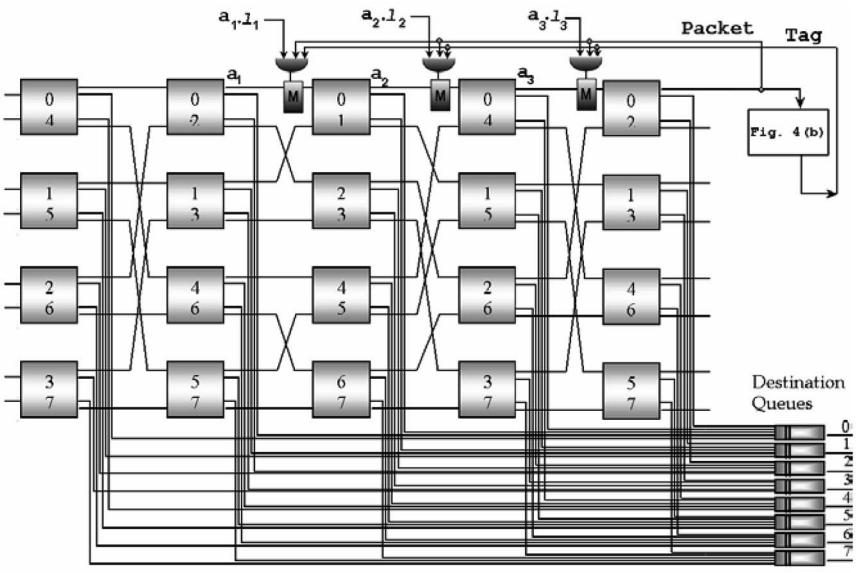

Fig. 5. ICO_8 with dynamic recirculating connections, $\mathrm{ICO}_8^{\mathrm{FO}}$.

when the total number of stages is smaller. Simulation results of $\mathrm{ICO}^{\mathrm{FA}}$ with different total stages will be presented in the next section. For the cases of buffered ICO (as will be treated in the next section), however, $\mathrm{ICO}_N^{\mathrm{FA}}$ is found to exhibit only negligibly better performance than $\mathrm{ICO}_N^{\mathrm{S}}$.

3.2.3 Dynamic Recirculating Connections—First "1" Bit

While $\mathrm{ICO}^{\mathrm{FA}}$ provides three entry points per recirculating connection and allows a packet to reenter the fabric at the first available stage, it does not check if the routing bit(s) corresponding to that reentry stage is (are all) "0." This is not the best point for the packet to reenter the fabric, unless the routing bit(s) for that stage is not (are not all) "0." The reason is two-fold: 1) the packet stays in the same row after that stage if the routing bit(s) is (are all) "0," and having the packet reenter the fabric there costs one extra cycle unnecessarily, and 2) the reentered packet may conflict with another packet in the reentry stage and get deflection-routed, taking an extra (otherwise unneeded) recirculating trip to "correct" this deflection-routing. The throughput gain out of $\mathrm{ICO}^{\mathrm{FA}}$ may be offset somewhat by an increased mean latency.

The best entry point for a recirculated packet obviously corresponds to the *first* "1" bit in its routing tag, dubbed $\mathrm{ICO}^{\mathrm{FO}}$. This is because the packet has to get "corrected" in that corresponding stage before it can reach its destined row. To this end, an entry point is provided at every stage of the last copy for each recirculating connection, as demonstrated in Fig. 5. The control signal for the entry point at stage i shown in the figure is given by $l_i = \underline{t}_1\ \underline{t}_2 \cdots \underline{t}_{i-1} t_i$, where $\underline{t}_j$ (or t_i) is the "NOR" (or "OR") result of the routing bit(s) for stage j, $1 \le j \le i-1$ (or stage i). If the reentered packet happens to conflict with another packet at the entry point, the former packet is dropped. This is done simply by making use of the availability indicator of an inlet at stage i of the last copy, a_i, as shown in Fig. 5. If the packet reenters the fabric via the entry point of stage f, its tag is to be rotated leftward by $(f-1) \times \log_2 b$ bit positions before reentry, using the same logic as depicted in Fig. 4b, with control signals produced using t_j rather than a_j, for $1 \le j \le 2$. After a packet reenters the fabric, it is routed in the same way as that described for $\mathrm{ICO}^{\mathrm{FA}}$. Unlike $\mathrm{ICO}_N^{\mathrm{FA}}$, $\mathrm{ICO}_N^{\mathrm{FO}}$ requires *exactly* $\log_b N$ entry points for each recirculating connection, since the first "1" tag bit may correspond to the last stage (of the last copy). As $\mathrm{ICO}^{\mathrm{FO}}$ selects the best entry point for each reentered packet, it is expected to exhibit the highest performance measures of interest for a given number of total stages, most suitable (among the three designs) for scalable router construction.

3.3 Buffered Switching Fabrics

Buffers are commonly introduced to the SEs of switching fabrics for performance improvement, in addition to their needs in packet switches for offering quality-of-service guarantees by establishing separate queues at each switch port for different traffic flows with various priority levels [19]. Buffers can be placed at the output (or input) ports of SEs to constitute output (or input) queues. Output queuing is known to exhibit better performance and be spared from the head-of-line blocking problem, but it often requires a "speedup" to achieve its peak throughput (of 100 percent potentially), where a speedup refers to that the switch core runs faster than the external links, so that multiple packets competing for an output port can be accepted by its associated output queue in one cycle [13]. Performance of crossbar-based packet switching under input and output queues has been studied extensively [7], [12], [13], with focuses chiefly on how to schedule packets and how to design an efficient queuing structure. In this work, we intend to explore the potential savings in total numbers of stages (of SEs) resulting from incorporating output queues in our proposed switching fabrics. This savings translates to cost reduction because the cost of a multistage-based fabric is largely proportional to the total number of stages (which dictates the overall chip count). In addition, few stages make the concentration logics in front of each destination queue simpler, further lowering the hardware cost.

Let the speedup at an output queue (of SEs) be denoted by ζ. For a fabric constructed out of $b \times 2b$ SEs, ζ is clearly no greater than b since there are at most b packets competing for an output queue (at an SE) in one cycle. While the number of lines terminating at each destination queue for a proposed fabric is equal to the number of stages in the fabric, a simple selector is placed in front of each destination queue in order to choose up to ζ packets in each cycle for acceptance by the queue, and those packets chosen are from the *last* ζ *active* stages (with respect to the destination), where an active stage has a packet to be delivered to the destination queue. More details about the selector design are provided in the next section. Destination queues are thus assumed to have a speedup of ζ in our switching fabrics. As will be illustrated by the simulation results later, performance of all the proposed fabrics is sensitive to ζ, and an increase in ζ leads to better performance for any given $\mathrm{ICO}_N^{\mathrm{S}}$, $\mathrm{ICO}_N^{\mathrm{FA}}$, or $\mathrm{ICO}_N^{\mathrm{FO}}$. When a fabric is composed of 4×8 SEs, it is possible to run at a rate of 200 MHz, which is deemed rather moderate with the current manufacturing technology (given that the spider chip which employs a full 6×6 crossbar is operating at 200 MHz [11]). With this clock rate, ζ may be pushed to 4, when each output queue takes one packet in 1.25 *ns.* This is realizable practically, since the current on-chip SRAM (static RAM) can have an access time as little as 1 *ns.*

The routing procedure in SEs with output queues is identical to that explained in Section 3.1 for SEs without queues, except for deflection-routing decision. Specifically, in a $b \times 2b$ SE with the speedup of $\zeta(\leq b)$ at output queues, if there are more than ζ packets heading for an output port in one cycle, the associated output queue can accept up to ζ such competing packets in a cycle, provided that the queue has capacity to hold them. Note that each queue has a specified capacity which can hold a number of packets (of fixed length). In each cycle, the packet, if any, at the head of each queue is moved to the next stage, say stage p, as follows: If the packet has reached its destined row in stage p, it is sent to the queue associated with the local port connected to its destination queue (referred to as a local queue), provided that the queue has capacity to take it and there are no more than ζ packets competing for the same queue at the same cycle. If the queue has no capacity left, the packet is sent to the queue associated with the remote output port connecting to the same row (called a remote queue) in stage *p+1*; the packet then attempts to reach its destined local queue in stage *p+1* during the next cycle. If there are more than ζ competing packets, ζ of them are randomly chosen for reaching the destined local queue, with the rest directed to the associated remote queue (for delivery to their destined local queue in the next stage).

If a packet has not reached its destined row in stage p, it is forwarded to a remote queue determined by the routing bit(s) of the packet. When there are more than ζ packets competing for the remote queue, the distance of each packet dictates if the packet is to be sent to the desired queue or to be deflection-routed: the packet which is closer to its destination is given priority, like in the nonbuffered cases. The priority resolution logic illustrated in Fig. 2b for nonbuffered SEs can be employed for this purpose after a minor modification. For the speedup of ζ, the SE core is operating ζ times faster. However, the resolution logic, being a simple combinational circuit, in an SE can arrive at one decision in less than $(1/\zeta)$ cycle time (called a phase); for example, with the cycle time of 5 *ns* and $\zeta = 2$, we have $(1/\zeta)$ cycle time equal to 2.5 *ns* and the resolution logic is expected to produce one decision in a fraction of 2.5 *ns*. After a packet is chosen as a winner, the logic input corresponding to the winner packet is changed to all "1" (reflecting a farthest distance) so that the packet with the second smallest distance is selected in the next decision phase. This process repeats ζ times in a cycle to choose ζ winners to reach their desired queues. All losers, if any, are deflection-routed to other remote queue(s) in the SE.

4 Fabric Design Details

This section deals with two design details related to the proposed switching fabrics in sequence: packet resequencing and packet selection at destination queues.

4.1 Packet Resequencing

With recirculating connections and deflection-routing, the three fabric designs under consideration, like the closed-loop Shuffleout (CSO), suffer from the packet out-of-sequence problem, which can be resolved by adopting a resequencing mechanism in the destination queues. A message is broken into fixed-sized packets for transmission, and those packets which belong to the same message have to be reconstructed at their destined LC (i.e., line card) before being processed therein further (according to the protocol of the outgoing connection). The switching fabrics may deliver packets to destined LCs in a different order than they originated from the source LCs and, thus, a certain mechanism must be provided to reorder the packets into their original messages, called the resequencing process. Resequencing can be done by software, but this is not suitable for high-performance router use because a software approach is simply too slow to keep pace with the traffic flows in such routers. A hardware solution is therefore required, and one hardware solution is described below. This resequencing hardware can be placed after each destination queue as a separate component or can share the same buffer of a destination queue. One such hardware is employed to support a communication pair between a primary input and a destination.

Assuming that packets of a given message are numbered by unique, continuous ids, and there are Ψ slots in a resequencing component, where each slot holds one packet (of fixed length) and $\Psi = 2^c$. A register is employed to keep the smallest id among all packets waiting to be received, called the LB register as depicted in Fig. 6. It is initialized with 0 when a new message starts to arrive. A register of c bits, called the OF (signifying offset) register, keeps the position of the slot waiting to be filled by the packet with the smallest id. Once the slot is filled, OF is advanced to the next unfilled slot, with the advanced amount added to the value of the LB register and all slots in between forwarded to the destination LC for further processing. After the two waiting packets arrive at slots 3 and 4 in Fig. 6a, for example, the OF register is advanced to 6 (the next waiting slot) and the LB register is added by 3, with packets in slots 3, 4, and 5 forwarded along for processing. The result is illustrated in Fig. 6b, where slots 3, 4, and 5 have been emptied for use by subsequent packets in the future. Any packet which may be accepted by this resequencing hardware

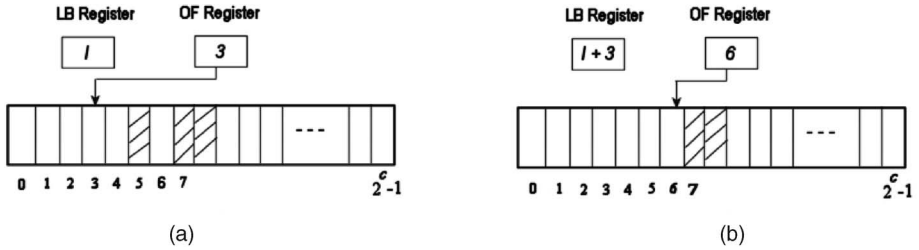

Fig. 6. (a) Resequencing hardware. (b) OF register advanced after slots 3, 4, and 5 are filled and forwarded.

must have its id falling between l and $(l + \Psi - 1)$ inclusively, where l is the value kept in the LB register, as shown in Fig. 6a. Let v be the id of an accepted packet. The slot for the packet to be placed is indexed by $i = (v - l + o) \bmod \Psi$, where o is the value held in the OF register and the mod function yields the c least significant bits of i. In essence, those Ψ slots in one resequencing hardware are viewed as a circular array, with the starting point recorded in the OF register.

4.2 Packet Selection at Destination Queues

A selector is placed in front of each destination queue to choose packets for terminating at the destination queue. For a fabric with k stages, there are k local links connecting to each selector, one from each stage. With the speedup of ζ, the selector chooses up to ζ packets in one cycle from those k inputs, where ζ is set to be the same speedup as that for SE output queues. Like the SE core, the selector operates ζ times faster than the links terminating at it. In each phase (defined as $1/\zeta$ cycle time), it chooses the rightmost "active" input among all k inputs to be received by the destination queue, as depicted in Fig. 7. At the end of each phase, the chosen input is reset (to become inactive using simple feedback lines as shown in the figure), ready for the next phase of selection. The selector repeats ζ times in one cycle to choose up to ζ packets, which are from its ζ rightmost "active" inputs. It functions in a way similar to what winner packets are chosen for an output queue in SEs among those competing packets, except that the queue in SEs chooses one packet in each phase based on the distance to a packet's destination (computed by a simple circuit depicted in Fig. 2b); once a packet is chosen for a queue in SEs, the distance of that packet is then set to the largest allowable value (i.e., all "1" to reflect the farthest distance) before the next phase starts.

5 Performance Evaluation and Results

The performance behaviors of proposed switching fabrics are evaluated by simulation. Different types of $b \times 2b$ SEs for building various fabric sizes (N) have been examined, including $b = 2, 4, 8$ and $N = 64,\ 256$. As the simulation results point to a similar trend, this section presents only the results of switching fabrics comprising 4×8 SEs with $N = 256$. We illustrate and discuss the results for nonbuffered switching fabrics first, followed by those for switching fabrics with output queues operating under different speedups. As will be seen, a buffered switching fabric requires much fewer stages to deliver a given performance level than its nonbuffered counterpart, even when the speedup is only two. The buffered switching fabrics under different traffic patterns possibly existing in practical settings are evaluated by simulation as well, with their results illustrated and compared.

5.1 Simulation Model

A copy of such a fabric consists of $\log_4(256) = 4$ stages. Each primary input is assumed to generate packets of fixed length randomly with their destinations uniformly distributed between 0 and 255 under different loads. A packet moves from one SE to another in the next stage in one cycle, based on the routing algorithm. A recirculated packet in any of the three fabric designs also takes one cycle to get

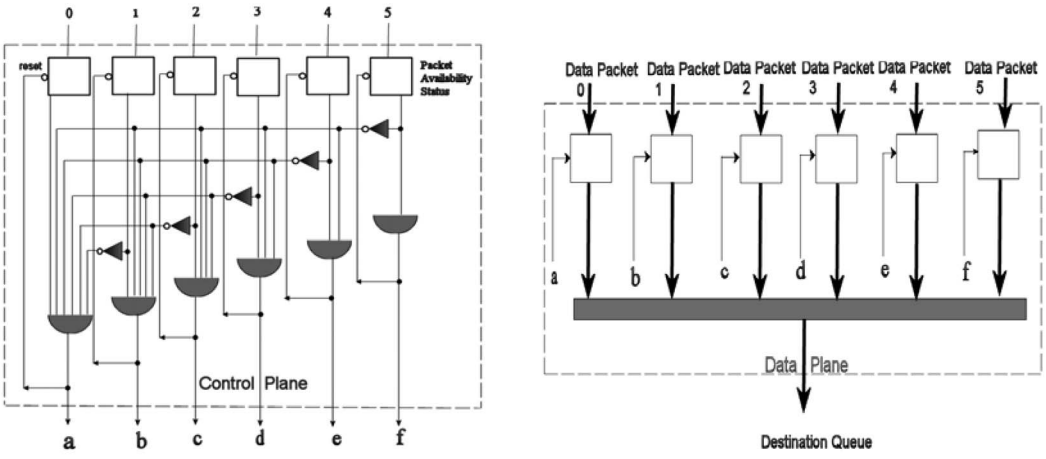

Fig. 7. A selector for switching fabrics with six stages.

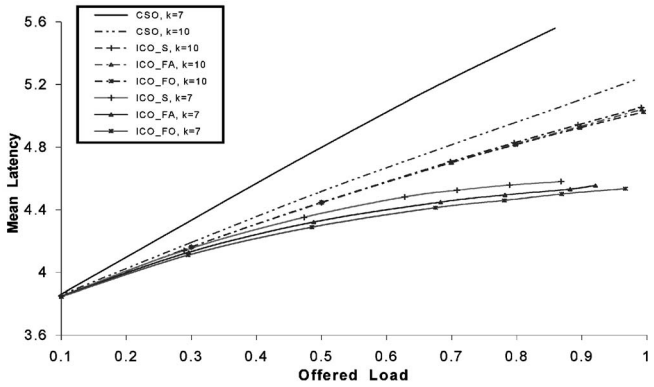

Fig. 8. Mean latency versus offered load for $k = 7$ and $k = 10$.

readmitted. Each data value given in Figs. 8, 9, 10, 11, 12, 13, 14, and 15 is the result after 200,000 clock cycles in our simulation, where this number of cycles is found to yield steady-state outcomes. The performance measures of interest include *mean latency*, *offered load* (or *throughput*), and *packet drop rate*. Mean latency signifies the average number of cycles for a packet to reach its destination queue after it is generated. The number of packets arriving at a typical destination queue per cycle is defined as offered load (or throughput), whereas the probability of a packet gets dropped in the switching fabric under a given load reflects the packet drop rate. For buffered switching fabrics, each SE output queue (either local or remote one) is equipped with a buffer for holding 12 packets. When an output queue runs out of its capacity (i.e., unable to keep those selected ζ packets in one cycle, where ζ is the speedup), some packets which otherwise should reach the queue are deflection-routed. Each queue has a bypass provision such that a packet entering the queue does not have to take 12 cycles to exit unless there are 11 packets situated in front of it. This provision allows a packet which enters an empty queue to leave the queue immediately in the next cycle. A small queue capacity degrades performance for the fabrics under consideration, in particular, when ζ is 4; on the other hand, an excessively large queue does not enhance performance noticeably. As a result, we only present results for the queue capacity equal to 12 slots (i.e., packets) here.

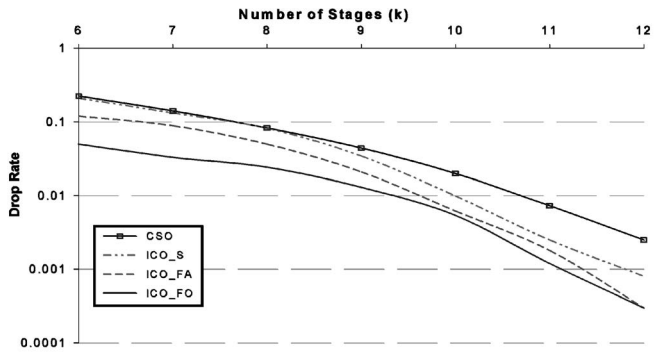

Fig. 9. Packet drop rate versus k under the packet generation rate of 1.0.

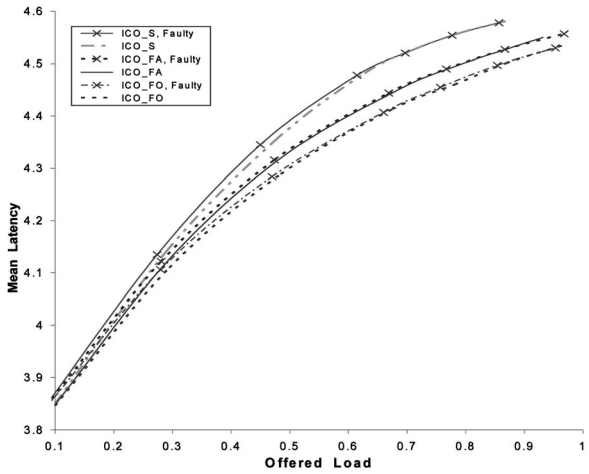

Fig. 10. Mean latency versus offered load under a single fault.

5.2 Results and Discussion of Proposed Switching Fabrics

Let k denote the total number of stages involved in a switching fabric. The mean latency versus offered load for $k = 7$ under $\mathrm{ICO}^{\mathrm{S}}_{256}$, $\mathrm{ICO}^{\mathrm{FA}}_{256}$, and $\mathrm{ICO}^{\mathrm{FO}}_{256}$ is depicted in Fig. 8. The performance results of a compatible closed-loop Shuffleout (denoted by CSO_{256}) are also included for comparison. As can be observed, $\mathrm{ICO}^{\mathrm{S}}_{256}$ under $k = 7$ exhibits considerably better performance than its CSO_{256} counterpart. For a given offered load, $\mathrm{ICO}^{\mathrm{S}}_{256}$ enjoys sizable reduction in mean latency, by up to almost 20 percent. The maximum sustained throughput is slightly higher under $\mathrm{ICO}^{\mathrm{S}}_{256}$ than under CSO_{256} for $k = 7$. This performance advantage is achieved, even though the constituent switching elements are less complicated in ICO than in CSO, as elaborated in Section 3.1. It results directly from the fact that reentered packets in ICO travel through the last four stages (i.e., last copy) where traffic is far lighter than that in earlier stages. When k grows, the performance gaps shrink, as can be found for the case of $k = 10$ in Fig. 8, since fewer packets then need to be recirculated.

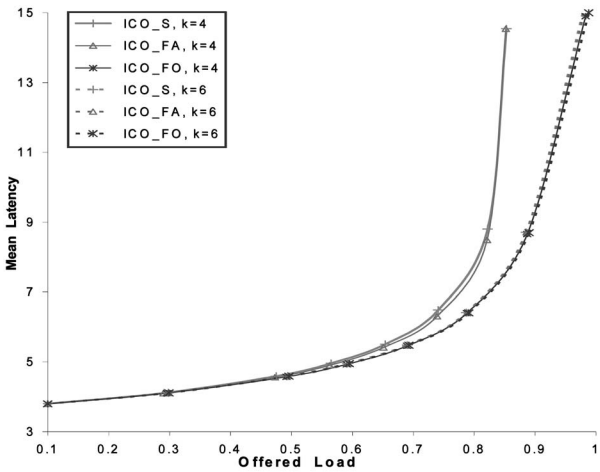

Fig. 11. Mean latency versus offered load for buffered switching fabrics under $k = 4$ and $k = 5$ with $\zeta = 2$.

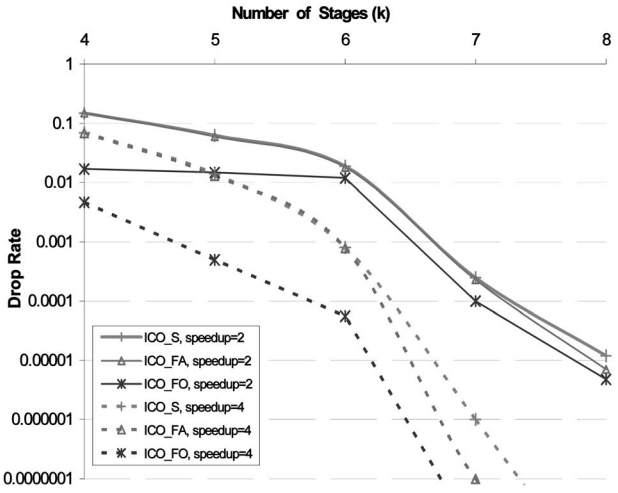

Fig. 12. Packet drop rate versus k for buffered switching fabrics under the load of 1.0.

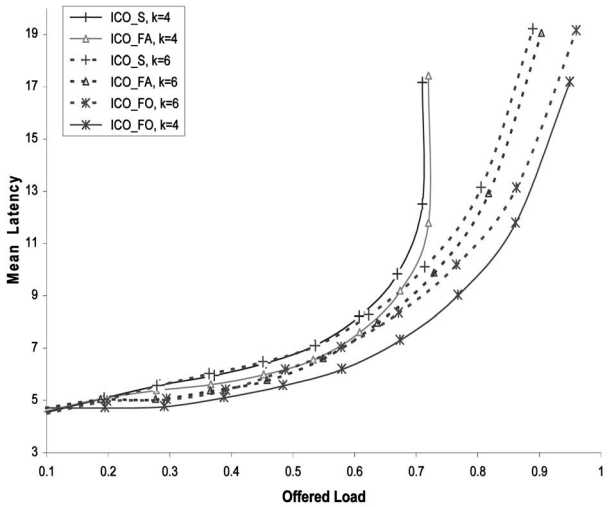

Fig. 14. Mean latency versus offered load for buffered switching fabrics with single hot ports of $\eta = 10$ percent under $k = 4$ and $k = 6$ with $\zeta = 2$.

The switching fabric of a router often employs a backpressure mechanism to refrain packets from entering the fabric once its resources are exhausted [15], [18], instead of dropping packets after they are admitted. When no backpressure mechanism is adopted, the drop rate performance measure serves as a good indicator of the probability that the backpressure mechanism is engaged. The packet drop rate versus k under the packet generation rate of 1.0 is demonstrated in Fig. 9. For any given k, $\mathrm{ICO}_{256}^{\mathrm{S}}$ is found to enjoy consistently a smaller drop rate than its CSO_{256} counterpart, as expected. Likewise, $\mathrm{ICO}_{256}^{\mathrm{FO}}$ outperforms $\mathrm{ICO}_{256}^{\mathrm{S}}$ and $\mathrm{ICO}_{256}^{\mathrm{FA}}$ for all k values examined. The gap between the drop rate of $\mathrm{ICO}_{256}^{\mathrm{FO}}$ and that of $\mathrm{ICO}_{256}^{\mathrm{S}}$ (or $\mathrm{ICO}_{256}^{\mathrm{FA}}$) is particularly large for $k < 8$. This is mainly because the switching fabric with $k < 8$ does not have a full copy of stages to route packets without competing with reentered ones (since less than two full copies of stages are involved), making it especially crucial to conserve fabric resources by admitting recirculated packets only through their best entry points, as done by $\mathrm{ICO}_{256}^{\mathrm{FO}}$. When k grows, fewer packets are recirculated and the performance gain due to $\mathrm{ICO}_{256}^{\mathrm{FO}}$ decreases accordingly, as shown in Fig. 9.

The recirculation designs have substantial impacts on overall performance, in particular, for a relatively small k, say, $k < 2 \times \log_b N$. When k equals 7, for example, $\mathrm{ICO}_{256}^{\mathrm{S}}$ achieves the maximum offered load of 0.87 only, whereas $\mathrm{ICO}_{256}^{\mathrm{FO}}$ can sustain the offered load of 0.97 (shown in Fig. 8). In addition, for any given throughput, $\mathrm{ICO}_{256}^{\mathrm{FO}}$ results in slightly lower mean latency than the other two designs, for both $k = 7$ and $k = 10$. From the performance standpoint, $\mathrm{ICO}_{256}^{\mathrm{FO}}$ is clearly superior. While $\mathrm{ICO}_{256}^{\mathrm{FO}}$ requires hardware for rotating the tag of every reentered packet properly along each recirculating connection (as described in Section 3.2), it takes fewer stages when compared with $\mathrm{ICO}_{256}^{\mathrm{S}}$ to achieve the same performance level. For example, $\mathrm{ICO}_{256}^{\mathrm{FO}}$ needs only seven stages to yield the maximum offered load achievable by $\mathrm{ICO}_{256}^{\mathrm{S}}$ with $k = 9$, according to Fig. 9. These savings in the reduced stage count and in simplifying interfaces to the destination queues more than offset added hardware since four pieces of tag

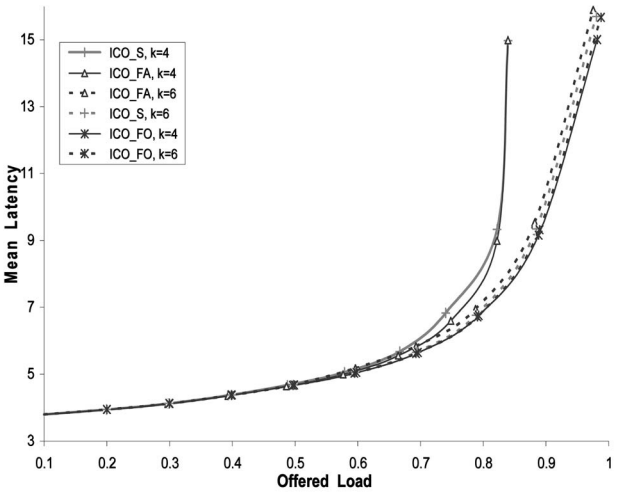

Fig. 13. Mean latency versus offered load for buffered switching fabrics with single hot ports of $\eta = 1$ percent under $k = 4$ and $k = 6$ with $\zeta = 2$.

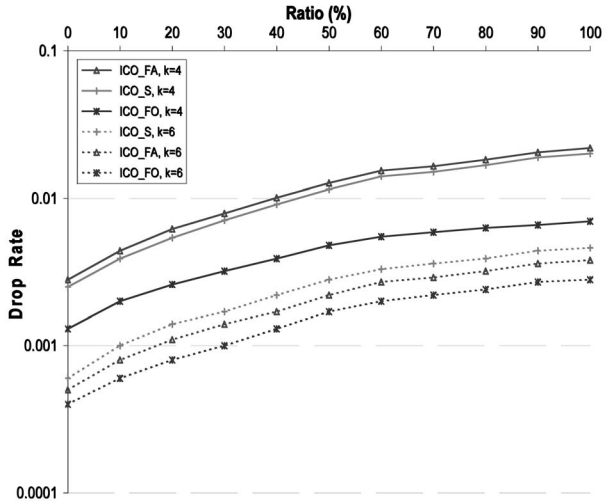

Fig. 15. Packet drop rate versus ratio of 40-Gbps links to 10-Gbps links (β) for $N = 256$.

rotation logics (for the outlets of an SE in the last stage) are far simpler than a 4×8 nonbuffered SE (consult Fig. 4b for a tag rotation logic). On the other hand, $\text{ICO}_{256}^{\text{FO}}$ possesses slightly more hardware than $\text{ICO}_{256}^{\text{FA}}$, but delivers a marked gain in the maximum offered load for $k = 7$.

As an SE failure has more detrimental effects on switching fabric performance than a link failure, we have evaluated the impacts of single faults by randomly designating one SE to be faulty. The faulty SE does not accept packets, so any packet which is to pass through the SE has to be deflection-routed in the prior stage. Mean latency versus offered load under a single SE failure in the second stage for $k = 7$ is depicted in Fig. 10, where the results under fault-free situations (given in Fig. 8) are also included for comparison. It is observed that a single SE failure leads to negligible performance degradation for all considered fabric designs, as they naturally can tolerate any single fault in the second stage, provided $k \geq 6$, because stage 2 and stage 6 in ICO_{256} "correct" the same tag bits. In general, ICO_{256} can tolerate any single fault in stage i of a copy, provided that $k \geq (4 + i)$, as ICO_{256} then contains at least two identical stages able to "correct" the tag bits corresponding to stage i. The results of single failures existing in another stage or for a different k have been gathered, and they all lead to the same conclusion. This capability of tolerating single faults without measurable performance degradation renders ICO ideally suitable for larger fabric construction, where a fault is increasingly likely.

5.3 Results of Buffered Switching Fabrics

A buffered switching fabric is expected to exhibit better performance than its nonbuffered counterpart for any given number of stages of SEs. In Fig. 11, mean latency versus offered load for the three buffered switching fabrics is demonstrated under $k = 4$ and $k = 6$ with the speedup of $\zeta = 2$. It should be noted that the minimum number of stages of 4×8 SEs required for any multistage-based fabric of size $N = 256$ is four (referring to as one copy of stages earlier). With $k = 4$, ICO^{S} exhibits the maximum offered load of 0.85, almost identical to what ICO^{FA} does. In addition, ICO^{S} and ICO^{FA} are found to yield roughly the same mean latency throughout the entire range of the offered load, signifying that they offer almost identical performance (despite more complicated recirculating connections for ICO^{FA}). On the other hand, ICO^{FO} reaches the maximum throughput exceeding 0.98 for $k = 4$, and its corresponding performance curve is noticeably lower than those of ICO^{S} and ICO^{FA}. This indicates that ICO^{FO} clearly outperforms its two counterparts when buffers are introduced to SEs, like nonbuffered cases demonstrated previously. If the number of stages grows to 6 (i.e., $k = 6$), ICO^{S} and ICO^{FA} again exhibit almost identical performance, with their maximum offered loads beyond 0.98, as illustrated in Fig. 11. Under this speedup (of $\zeta = 2$), it appears that $k = 6$ is adequate for ICO^{S} and ICO^{FA} to sustain satisfactory performance levels. As expected, ICO^{FO} again exhibits better performance than ICO^{S} and ICO^{FA}, but the performance gap shrinks in this case compared with the case of $k = 4$. The maximum offered load for ICO^{FO} with $k = 6$ approaches 0.99. For ICO^{S} and ICO^{FA}, the increase of k from four to six substantially enhances its performance, whereas for ICO^{FO}, the performance improvement amount is negligible since its performance under $k = 4$ is very good already. In general, $\text{ICO}_{N}^{\text{FO}}$ is a superior design choice among the three buffered switching fabrics (composed of $b \times 2b$ SEs), particularly if k equals, or is only slightly more than, $\log_b N$.

The packet drop rate as a function of k is shown in Fig. 12 for the three switching fabrics with different speedups (ζ) under the load of 1.0. Under a speedup of $\zeta = 2$, all switching fabrics experience reduced drop rates as k grows. The drop rate reduction accelerates when k goes beyond six, because the recirculated packets not only are few, but also are fed back into the fabrics through a stage whose traffic is expected to be very light. For any k, buffered ICO^{FO} always maintains a smaller drop rate than its ICO^{S} and ICO^{FA} counterparts. If the speedup is pushed up aggressively to $\zeta = 4$, the drop rates for the three switching fabrics not only are consistently much lighter for any k, but also go down quicker as k increases, when compared with those under $\zeta = 2$. As might be expected, ICO^{FO} is observed to outperform its two counterparts by a wide margin. Under $k = 6$, for example, ICO^{FO} gives rise to a drop rate less than 10^{-4}, in contrast to 10^{-3} exhibited by ICO^{S} and ICO^{FA}. When k exceeds six, ICO^{FA} starts to outperform ICO^{S} noticeably and the drop rate gap between them expands as k grows further.

Comparing these results with those of nonbuffered cases reveals that the introduction of buffers to SEs yields considerable performance enhancement for a given k. As an instance, ICO^{S} with $k = 6$ delivers a maximum offered load of 0.79 (from Fig. 9) without buffers in SE's, as opposed to more than 0.98 with buffers. Likewise, ICO^{S} under $k = 6$ arrives at the maximum offered load of 0.95 without buffers, in contrast to 0.99 with buffers. Alternatively, the switching fabrics with buffers need far fewer stages to achieve a given performance level than their compatible nonbuffered ones. For example, the drop rate of our buffered ICO^{FO} (or ICO^{S}) with $k = 6$ is smaller than that of its nonbuffered counterpart with $k = 9$ from Fig. 9, reflecting a significant savings in the total number of SEs involved. While an SE with buffers is more complex than a corresponding SE without buffers, the overall cost of a switching fabric is likely to depend mostly on the number of stages (or SEs), because the number of stages dictates the chip count of a fabric. This reduction in the number of stages also translates to a simpler circuit (i.e., selector) for terminating packets at each destination queue. A buffered switching fabric thus appears more attractive than its nonbuffered counterpart due to its potentially lower cost.

The impact of a single SE failure on the performance of buffered switching fabrics was measured via simulation as well under different stage numbers (k) and speedups (ζ). Since a desirable k is normally chosen to be $\log_b N$ plus a small constant, say from zero to two, for a buffered $\text{ICO}_{N}^{\text{FO}}$ composed of $b \times 2b$ SEs when ζ is two, the impact of an SE failure is more pronounced than its nonbuffered counterpart (whose k is often $2 \times \log_b N$ or larger). For example, $\text{ICO}_{256}^{\text{FO}}$ with $k = 4$ experiences an 8 percent more packet drop rate if a failure arises in the last stage (when compared with a fault-free case), under the load of 1.0. If the failed SE happens to an earlier stage of the fabric, the increase in the drop rate rises swiftly. When a failure is at the second stage of $\text{ICO}_{256}^{\text{FO}}$, the drop rate increases by as much as 85 percent (if the load equals 1.0). This higher drop rate results from

the fact that ICO_{256}^{FO} with $k = 4$ (containing just one copy of stages) has no alternative SE in existence in the fabric to take the role of a failed SE as far as packet routing is concerned, and all packets which rely on the failed SE to correct their tags before reaching their destinations are dropped altogether. When k grows, the impact of a singe failure is reduced rapidly.

5.4 Results under Different Traffic Patterns

Traffic over switching fabrics is assumed uniform so far in our simulation study. This section deals with buffered ICO under different traffic patterns. Switching fabrics in typical routers often carry nonuniform traffic in practical situations, resulting from the fact that a router usually designates one or a few ports as the *default ports* for those packets whose destinations cannot match any entry in the routing/forwarding tables, to travel through in order to reach more powerful routers with larger and more complete routing/forwarding tables. Note that packets at a router are transported from their arrival LCs (line cards) to their departure LCs over its employed switching fabric, based on the table lookup results using packet destinations as lookup keys. Traffic over switching fabrics is thus often nonuniform, with one or several ports acting as the *hot* port(s) which take unproportionally high traffic rates. To measure ICO performance under various nonuniform traffic patterns, we considered the cases of single hot ports and multiple hot ports, with hot ports at different locations which collectively receive η percent ($0 \leq \eta \leq 100$) hot traffic in addition to their fair shares of regular traffic. Specifically, for a given η and a packet generation rate λ (i.e., load) under a single hot port, the hot port receives $\lambda \times \eta$ percent hot traffic from every primary input plus $(\lambda \times (1 - \eta\%)/N) \times N = \lambda(1 - \eta\%)$ regular traffic from all primary inputs combined, where N is the fabric size. Under h (> 1) multiple hot ports, hot traffic is shared equally by those hot ports, namely, each hot port receives $(\lambda \times \eta\%)/h$ hot traffic from every primary input plus regular traffic.

For a given η, traffic congestion in ICO, if existing, is more pronounced under a single hot port than under multiple hot ports. As a result, we demonstrate simulation results for single hot ports only. It is found through our extensive simulation that ICO performance is *insensitive* to the hot port position for any η value. Mean latency versus offered load for buffered switching fabrics under k = 4 and k = 6 with $\zeta = 2$ is depicted in Fig. 13, where the hot port is chosen at port 0 and η equals 1 percent (which is likely to reflect the scenario of a backbone router whose routing table is normally large enough to yield matching for almost all packet destination, leading to a very small fraction of traffic delivered to the default (hot) port. As can be observed in this figure, buffered ICO^{FO} cases excel their counterparts for both $k = 4$ and $k = 6$, like the uniform traffic situation demonstrated in Fig. 11. Specifically, ICO_4^{FO} achieves the maximum offered load of 0.98 while ICO_4^S exhibits the maximum offered load of 0.84 only, in the presence of a single hot port with η = 1 percent. On the other hand, ICO_6^{FO} and ICO_6^S give rise to roughly the same maximum offered load (of 0.99 and 0.98, respectively). When compared with their corresponding results under uniform traffic depicted in Fig. 11, the results in Fig. 13 indicate that ICO can handle hot port traffic with little performance degradation.

It is expected that a bigger η will lead to more severe congestion at the hot port and, thus, lower performance, as can be seen in Fig. 14, where η is 10 percent (which may better reflect the case of edge routers, or equivalently, called metropolitan routers by the industry). This hot port, now experiencing a very high packet rate (even under the offered load of 0.1 when the packet rate to the hot port is $256 \times (0.1 \times 10\%) + 0.1 \times (1 - 10\%) = 2.65$), causes noticeable performance degradation. As an example, mean latency under offered load = 0.1 increases to 4.5 cycles (or more) from about 3.8 cycles seen for η = 1 percent shown in Fig. 13. Similarly, the maximum offered load drops to 0.71 (or 0.89) from 0.84 (or 0.98) for ICO_4^S (or ICO_6^S), and it is down slightly for ICO^{FO}. Under this heavy hot port rate, ICO_4^{FO} exhibits better performance than ICO_6^{FO}, as opposed to that ICO_6^{FO} always outperforms ICO_4^{FO} under uniform traffic (shown in Fig. 11) and light nonuniform traffic (shown in Fig. 13). This is mainly because a saturation tree has then developed across ICO with its root at the hot port and those packets destined for the hot port experiences extremely high latencies that push the mean latency up more noticeably in ICO_6^{FO} (which has 50 percent more stages to hold those hot-port packets longer) than in ICO_4^{FO}. This saturation tree phenomenon was well-known in a multistage network under hot-spot traffic, and it tends to degrade performance considerably [22]. Compared with ICO_4^{FO}, ICO_6^{FO} exhibits a limited increase in the offered load (for any given packet generation rate), but experiences a big leap in the mean latency, leading to its inferior performance. This situation was not observed for ICO_6^S (or ICO_6^{FA}) in comparison to its 4-stage counterpart, since ICO_6^S (or ICO_6^{FA}) achieves a far larger maximum offered load, up to 0.84 (or 0.86), and is able to yield a higher offered load for any packet generation rate $\lambda > 0.6$.

Switching fabrics are employed to interconnect key components in routers, like routing engines and line cards, whose ports terminate external links connected to neighboring routers. The fastest link speed for commercially available routers is 10 Gbps now and is expected to rise to 40 Gbps soon. As IP traffic accounts for the vast majority of current Internet traffic, we consider routers carrying such traffic here. An IP packet is of variable length ranging from 40 bytes up to 64K bytes, and it is fragmented into *data units* of fixed length in switching fabrics for transmission. The clock rate of a fabric normally is so chosen that a data unit can be transmitted in one cycle time from an SE to its connected SE in the next stage. Assume that the data unit selected for ICO is of **40 bytes** (i.e., shortest possible IP packet length) and the clock cycle time is **5 *ns*** (i.e., SEs operating at the rate of 200 MHz like the spider chip [9]), meaning that ICO is able to move a data unit along a link from one SE to a connected SE (or to a destination queue) in 5 *ns*. While the data rates of external router links typically vary in a wide range, dictated by the port interfaces adopted, we consider ICO used in a router for connecting external links of 10 Gbps and 40 Gbps only (through different LCs) since lower input data rates lead to lighter traffic over ICO and, thus, better performance. The simulation results for various

ICO configurations of size 256 with various percentages of 40-Gbps input links are demonstrated in Fig. 15, where $(\beta\% \times 256)$ primary inputs are of 40-Gbps, with the rest being 10-Gbps. A 40-Gbps input link translates to a packet generation rate of (40 Gbps $\times$ 5 ns/cycle) / (40 bytes/packet $\times$ 8 bits/byte) = 0.625 packet/cycle, and those 40-Gbps links are randomly chosen (among 256 primary input links). As expected, the packet drop rate grows monotonically for a larger β since heavier traffic then exists in the fabric, and ICO_6 always outperform their 4-stage counterparts, with $\mathrm{ICO}_6^{\mathrm{FO}}$ yielding best performance. When β equals 100, for example, $\mathrm{ICO}_6^{\mathrm{FO}}$ has a drop rate of 0.0028 only, in comparison to 0.0046 of $\mathrm{ICO}_6^{\mathrm{S}}$ and 0.007 of $\mathrm{ICO}_4^{\mathrm{FO}}$. $\mathrm{ICO}_6^{\mathrm{FO}}$ is able to handle traffic in a large router with high-speed line cards housing even 40-Gbps ports, readily applicable to scalable routers in the future.

6 Conclusion

While switching fabrics with centralized scheduling like shared buses and crossbars [17] are commonly adopted in existing commercial routers, their scalability is rather limited, rendering it infeasible to include large numbers of LCs (line cards) in such a router. In this article, scalable switching fabrics based on a multistage structure with different recirculating designs have been introduced, referred to as $\mathrm{ICO}^{\mathrm{S}}$, $\mathrm{ICO}^{\mathrm{FA}}$, and $\mathrm{ICO}^{\mathrm{FO}}$, respectively. With distributed routing, they aim to interconnect large numbers of LCs in high-performance routers. These fabrics employ far simpler switching elements (SEs) than an earlier multistage-based switch, known as the closed-loop Shuffleout (CSO), because no distance computing logics are needed. They all outperform a compatible CSO, resulting from recirculating packets to the last copy of stages where traffic is far lighter. In particular, $\mathrm{ICO}^{\mathrm{FO}}$ is demonstrated to prevail among all the fabric designs, made possible by employing simple logics to recirculate packets through best entry points in the last copy (of stages) without wasting resources or causing unnecessary conflicts. For example, $\mathrm{ICO}_{256}^{\mathrm{FO}}$ with seven stages (i.e., $k = 7$) of 4×8 SEs can deliver the performance level of $\mathrm{ICO}_{256}^{\mathrm{S}}$ with nine stages, leading to considerable savings in hardware complexity. The savings are attributed in part to a reduced stage count and in part to simplified selectors in front of destination queues, each of which is terminated by one SE in every stage.

When buffers are incorporated in SEs to create a queue for each output port, the number of stages (of SEs) required for a proposed switching fabric to achieve a given performance level is lowered significantly when compared with its nonbuffered counterpart, even for a small speedup of two. This reduction in the number of stages also leads to a simpler logic for terminating packets at each destination queue. As a result, buffered switching fabrics seem to have lower costs than compatible nonbuffered ones, because the overall cost of a fabric is determined largely by the stage count (which affects the chip count). Our simulation results reveal that buffered $\mathrm{ICO}_N^{\mathrm{S}}$ and $\mathrm{ICO}_N^{\mathrm{FA}}$ exhibit roughly identical performance for any given k and ζ (speedup), but buffered $\mathrm{ICO}_N^{\mathrm{FO}}$ clearly prevails, in particular, when k is close to $\log_b N$ and ζ is small (say, two in $b \times 2b$ SEs). Buffered switching fabrics are evaluated under different traffic patterns and are demonstrated to effectively handle nonuniform traffic (in the presence of hot ports) and varying packet rates at primary inputs for future routers.

While single failures at the links or constituent switching elements (SEs) of nonbuffered fabrics can be tolerated naturally with negligible performance degradation (since they contain $2 \times \log_b N$ or more stages of SEs), they have more pronounced impacts on the performance of buffered fabrics, which often involve much fewer stages. A buffered switching fabric is not fault-tolerant, for it contains fewer than two copies of stages. While buffered $\mathrm{ICO}^{\mathrm{S}}$, $\mathrm{ICO}^{\mathrm{FA}}$, and $\mathrm{ICO}^{\mathrm{FO}}$ all possess good scalability and low overall costs, buffered $\mathrm{ICO}^{\mathrm{FO}}$ is especially suitable for large sized construction, because it has the smallest stage count to guarantee a given performance level, in particular, under speedup = 2.

Acknowledgments

The author would like to thank Kiran Ponnuru, Kemathat Vibhatavanij, and Ravi C. Batchu for their assistance in developing the simulators for nonbuffered and buffered switching fabrics and in preparing figures used in this article. This work was supported in part by the US National Science Foundation under Grants EIA-9871315 and CCR-0105529, and by the Army Research Office under Grant/Cooperative Agreement No. DAAG55-98-1-0240. A preliminary version of this work was presented at the 2002 International Conference on Parallel Processing, August 2002.

References

[1] S. Bassi et al., "Multistage Shuffle Networks with Shortest Path and Deflection Routing for High Performance ATM Switching: The Open-Loop Shuffleout," *IEEE Trans. Comm.*, vol. 42, pp. 2881-2889, Oct. 1994.

[2] M. Decina, P. Giacomazzi, and A. Pattavina, "Multistage Shuffle Networks with Shortest Path and Deflection Routing for High Performance ATM Switching: The Closed-Loop Shuffleout," *IEEE Trans. Comm.*, vol. 42, pp. 3034-3044, Nov. 1994.

[3] S. Blake et al., "An Architecture for Differentiated Services," Internet IETF RFC 2475, Dec. 1998.

[4] A. Botta et al., "The 16×622 Mbits/s COM16M: The PRELUDE Switch Architecture Integrated into a 6-Million Transistor Monochip," *Proc. 22nd European Solid-State Circuits Conf.*, Sept. 1997.

[5] H. Chan, H. Alnuweiri, and V. Leung, "A Framework for Optimizing the Cost and Performance of Next-Generation IP Routers," *IEEE J. Selected Areas in Comm.*, vol. 17, pp. 1013-1029, June 1999.

[6] N. Christin, J. Liebeherr, and T. Abdelzaher, "A Quantitative Assured Forwarding Service," *Proc. IEEE INFOCOM'02*, pp. 864-873, June 2002.

[7] S. Chuang et al., "Matching Output Queuing with a Combined Input Output Queued Switch," *IEEE J. Selected Areas in Comm.*, vol. 17, pp. 1030-1039, June 1999.

[8] Cisco Systems, Cisco 12016 Gigabit Switch Router, Data Sheet, http://www.cisco.com, 2001.

[9] M. Decina, P. Giacomazzi, and A. Pattavina, "Shuffle Interconnection Networks with Deflection Routing for ATM Switching: the Open-Loop Shuffleout," *Proc. 13th Int'l Teletraffic Conf.*, pp. 27-34, June 1991.

[10] N. Endo et al., "Shared Buffer Memory Switch for an ATM Exchange," *IEEE Trans. Comm.*, vol. 41, pp. 237-245, Jan. 1993.

[11] M. Galles, "Spider: A High-Speed Network Interconnect," *IEEE Micro*, vol. 17, pp. 34-39, Jan./Feb. 1997.

[12] A.L. Gupta and N.D. Georganas, "Analysis of a Packet Switch with Input and Output Buffers and Speed Constraints," *Proc. IEEE INFOCOM'91*, pp. 694-700, Apr. 1991.

[13] M.G. Hluchyj and M.J. Karol, "Queueing in High-Performance Packet Switching," *IEEE J. Selected Areas in Comm.,* vol. 6, pp. 1587-1597, Dec. 1988.
[14] M. Katevenis, P. Vatsolaki, and A. Efthymiou, "Pipelined Memory Shared Buffer for VLSI Switches," *Proc. ACM SIGCOMM'95,* pp. 39-48, Aug. 1995.
[15] M. Katevenis, D. Serpanos, and E. Spyridakis, "Switching Fabrics with Internal Backpressure Using the ATLAS I Single-ChipATM Switch," *Proc. IEEE GLOBECOM'97,* pp. 242-246, Nov. 1997.
[16] N.F. Maxemchuk, "Comparison of Deflection and Store-and-Forward Techniques in the Manhattan Street and Shuffle-Exchange Networks," *Proc. IEEE INFOCOM'89,* pp. 800-809, Apr. 1989.
[17] N. McKeown, "The *i*SLIP Scheduling Algorithm for Input-Queued Switches," *IEEE/ACM Trans. Networking,* vol. 7, no. 2, pp. 188-201, Apr. 1999.
[18] N. McKeown et al., "The Tiny Tera: A Packet Switch Core," *IEEE Micro,* vol. 17, pp. 26-33, Jan./Feb. 1997.
[19] S. Moon, J. Rexford, and K.G. Shin, "Scalable Hardware Priority Queue Architectures for High-Speed Packet Switches," *IEEE Trans. Computers,* vol. 49, pp. 1215-1227, Nov. 2000.
[20] A. Pattavina, "Performance Evaluation of Batcher-Banyan Interconnection Networks with Output Pooling," *IEEE J. Selected Areas of Comm.,* vol. 9, pp. 95-103, Jan. 1991.
[21] M. Pease, III, "The Indirect Binary *n*-Cube Microprocessor Array," *IEEE Trans. Computers,* vol. 6, no. 5, pp. 250-265, May 1977.
[22] G. Pfister and V. Norton, "'Hot-Spot' Contention and Combining In Multistage Interconnection Networks," *IEEE Trans. Computers,* vol. 34, no. 10, pp. 943-948, Oct. 1985.
[23] PMC-Sierra Inc., TT1 Chipset, Data Sheet, http://www.pmc-sierra.com, 1999.
[24] K.J. Schultz and P.G. Culak, "CAM-Based Single-Chip Shared Buffer ATM Switch," *Proc. IEEE Int'l Conf. Comm. (ICC),* pp. 1190-1195, June 1994.
[25] I. Stoica and H. Zhang, "Providing Guaranteed Services without Per-Flow Management," *Proc. ACM SIGCOMM'99,* pp. 81-94, Aug. 1999.
[26] F. Tobagi, T. Kwok, and F. Chiussi, "Architecture, Performance, and Implementation of the Tandem Banyan Fast Packet Switch," *IEEE J. Selected Areas in Comm.,* vol. 9, pp. 1173-1193, Oct. 1991.
[27] N. Tzeng and T. Darwish, "Implementation of Scalable Switches for Wireless Communications," technical report, Center for Advanced Computer Studies, Univ. of Louisiana at Lafayette, 2001.
[28] N. Tzeng, K. Ponnuru, and K. Vibhatavanij, "A Cost-Effective Design for ATM Switching Fabrics," *Proc. 1999 IEEE Int'l Conf. Comm. (ICC),* pp. S37.4.1-5, June 1999.
[29] W. Wang, K. Yang, and T. Lin, "A Terabit Switch Fabric with Integrated High-Speed CMOS Transceivers," *Proc. Eighth Symp. High-Performance Interconnects (Hot Interconnects 8),* Aug. 2000.
[30] D. Weil et al., "A 16×622 Mb/s ATM Switch: PRELUDE Switch Architecture Integrated into a 6-Million Transistor Monochip," *IEEE J. Solid-State Circuits,* vol. 32, pp. 1108-1114, July 1997.
[31] T. Wolf and J. Turner, "Design Issues for High-Performance Active Routers," *IEEE J. Selected Areas in Comm.,* vol. 19, no. 3, pp. 404-409, Mar. 2001.
[32] P. Wong and M. Yeung, "Design and Analysis of a Novel Fast Packet Switch—Pipeline Banyan," *IEEE/ACM Trans. Networking,* vol. 3, no. 1, pp. 63-69, Feb. 1995.
[33] H. Yamanaka et al., "Scalable Shared-Buffering ATM Switch with a Versatile Searchable Queue," *IEEE J. Selected Areas in Comm.,* vol. 15, pp. 773-784, June 1997.
[34] K. Yun, "A Terabit Multi-Service Switch," *IEEE Micro,* vol. 21, pp. 58-70, Jan./Feb. 2001.

Nian-Feng Tzeng received the PhD degree in computer science from the University of Illinois at Urbana-Champaign in 1986. He is currently a professor with the Center for Advanced Computer Studies, University of Louisiana at Lafayette, which he joined in 1987. His current research interest is in the areas of computer communications and networks, high-performance computer systems, parallel and distributed processing, and fault-tolerant computing. He was on the editorial boards of the *IEEE Transactions on Parallel and Distributed Systems* from 1998-2001 and of the *IEEE Transactions on Computers* from 1994-1998. He served as coguest editor of a special issue of the *Journal of Parallel and Distributed Computing* on distributed shared memory systems, September 1995, and as a Distinguished Visitor of the IEEE Computer Society, 1994-1997. He was the chair of Technical Committee on Distributed Processing, the IEEE Computer Society, from 1999-2002. He has been on the technical program committees of various conferences and will serve as the technical program chair of the 10th International Conference on Parallel and Distributed Systems, July 2004. Dr. Tzeng is a member of the ACM and a senior member of the IEEE and the IEEE Computer Society. He is the recipient of the outstanding paper award of the 10th International Conference on Distributed Computing Systems, May 1990, and received the University Foundation Distinguished Professor Award in 1997.