Cost-Effective Switching Fabrics with Distributed Control for Scalable Routers

Nian-Feng Tzeng and **Malcolm Mandviwalla**

Center for Advanced Computer Studies
University of Louisiana at Lafayette
Lafayette, LA 70504, USA

Abstract

*This paper deals with scalable switching fabrics for high-performance routers with large numbers of ports for connecting external links operating at various speeds to arrive at aggregate rates up to multi-terabits per second. The proposed switching fabrics employ no centralized scheduling and consist of small routing units (RU's), which are interconnected by multistage-based connecting components (CC's) in accordance with grid structures, with routing decisions made by RU's and CC's individually in a simple, distributed manner. They are referred to as **grid**-oriented, **multistage**-connected **RU**'s, dubbed **GMR**. With distributed routing, GMR enjoys good scalability and low hardware complexity. It is found, based on our extensive simulation, that GMR outperforms not only their crossbar counterparts for small sizes, but also their compatible designs aiming at large sized construction (built from multiple stages of small crossbars), despite its lower hardware complexity. Two types of chips are sufficient to permit any sized construction; one for RU's and another for CC's. The proposed switching fabrics are cost-effective, readily suitable for scalable routers.*

I. Introduction

High-performance routers able to handle large numbers of links will be necessary soon to accommodate continuing expansion of the Internet. Such a router has to scale well, perhaps to the aggregate processing performance range of multi-terabits per second [14]. It employs a switching fabric to interconnect key components like line cards (LC's), and the scalability of a router is dictated mainly by the switching fabric employed, for any given LC processing/forwarding capability. Current routers available commercially utilize crossbars (e.g., Cisco 12000 Series routers [5]) or shared busses (e.g., the Juniper M160 router) for their switching fabrics, permitting to interconnect small amounts of system components.

This work was support in part by the National Science Foundation under Grants CCR-9803505, EIA-9871315 and CCR-0105529, by the Army Research Office under Grant/Cooperative Agreement No. DAAG55-98-1-0240, and by the Board of Regents of the State of Louisiana under Contract No. LEQSF(2000-2001)-ENH-TR-90.

A multi-terabit router may connect hundreds of links, which are terminated at its LC's. While an LC typically takes multiple (slower) links for speed aggregation (e.g., both Cisco routers and the Juniper router allow up to 4 links per LC with an aggregate speed limited to 10 Gbps), the number of LC's in such a router can still be large, well beyond what the crossbars or the shared buses can deal with. This paper considers novel switching fabrics for routers with high scalability. Such a fabric comprises small routing units (RU's) interconnected by connecting components (CC's) in accordance with grid structures, where a CC is composed of a multistage interconnect. The 2-D grid structure is sufficient for constructing practical sized fabrics, where each RU is connected to one or multiple copies of CC's along the *x*-direction and the *y*-direction.

The performance of our proposed fabrics is determined by two design parameters: (1) the speedup of RU's and (2) the ratio of the CC speed to the RU speed, where the RU speedup signifies the number of competing packets which can be moved to an RU output queue in one cycle, and the speed ratio is larger than 1 (since an RU is far more complicated and runs slower than a constituent 4×4 crossbar of CC's). A larger RU speedup leads to a better performance level, so does a higher speed ratio. For any performance level desired to achieve, the RU speedup and the speed ratio are small, often no more than 3, if each RU is connected to 2 copies of CC's along both the *x*-direction and the *y*-direction. The proposed switching fabric routes packets (of fixed length) in a distributed manner and exhibits better performance than compatible designs targeted for ATM (asynchronous transfer mode) switches, despite its far lower hardware complexity. For a small size, it also enjoys better performance and yet less hardware complexity than its state-of-the-art crossbar counterpart. The proposed switching fabric needs merely two types of chips for constructing any sized switching fabric, leading to low costs and ideally suitable for scalable routers.

II. Related Work

This section outlines the general architecture of high-performance routers, followed by a brief description of two current switching fabric designs.

A. High-performance routers

All recent high-performance routers include multiple forwarding engines (FE's), which carry out forwarding table lookups simultaneously. For example, Cisco 12000 Series routers each admit up to 15 FE's plus one routing engine [4], whereas the Juniper M160 backbone router contains 4 FE's. A router connects external links through its ports, and a few ports are often housed in one LC. For a distributed router [2], each LC includes an FE so that table lookups can be done locally (e.g., Cisco routers). On the other hand, a parallel router hosts an array of FE's, which are separated from, and shared by, its LC's (e.g., Juniper routers).

As the router scales, more LC's are added and the switching fabrics employed to interconnect its constituent LC's grow accordingly. The switching fabrics for Cisco 12000 Series routers are crossbar-based, similar to the Tiny Tera [8]. The Juniper M160 router uses a shared bus for its switching fabric. For multi-terabit routers, with now commercially available links, the number of LC's needed can be in the hundreds, making it highly desirable to use switching fabrics with better scalability than what the shared bus or the crossbar may offer [14].

Note that one may choose to limit the size of switching fabrics to 32 and follows the crossbar configuration with centralized cell scheduling for terabit router construction, like the Yuni switch architecture [16] or the switch fabric design based on CMOS transceivers [13]. In this way, to scale up router performance translates to an increasing aggregation degree at the LC's, 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 routers. 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 processing/forwarding rate and thus router scalability.

B. Current switching fabrics

A compact switching fabric with an aggregate bandwidth of 320 Gbps, called the Tiny Tera, has been presented [8]. It is suitable for the core of a router, and a similar design was adopted for switching fabrics in the Cisco routers. The Tiny Tera is a single-stage switch, composed of one single crossbar with a central scheduler. At the beginning of each cycle time, the scheduler examines the contents of all input queues, decides upon the configuration of the crossbar, and chooses a set of conflict-free connections between inputs and outputs. The scheduling decision is passed back to the crossbar for setting up its configuration [8]. Being a centralized control, the Tiny Tera scales up to 32 ports only and carries out one scheduling in 40 *ns* (i.e., operating at a clock rate of 25 MHz as in the implementation of the PMC-Sierra Inc.'s TT1 Chipset [10] used for its construction).

For the past decade, various switching fabrics have been proposed for constructing scalable ATM switches. Among them, the Shuffleout is shown to demonstrate superior performance and good scalability [1]. The Shuffleout switch consists of multiple stages of unbuffered switching elements (SE's), which are small crossbars interconnected by a shuffle connection pattern between two adjacent stages. An SE, say, of size b×2b has b switch outlets terminating at b output queues (which are fed by links, one from each stage) to hold cells (i.e., fixed-length packets) as soon as they reach their destinations. In a Shuffleout with size 8 and b = 2, there are four SE's in each stage and each SE has two switch outlets terminating at two output queues. On receiving a cell, the SE always attempts to route the cell along the shortest path to the destined switch outlet. If two cells at an SE require the same switch outlet to an SE in the next stage (or to an output queue), deflection routing is applied so that the cell closer to its destined switch outlet is routed along the shortest path, whereas the other cell is deflection-routed [1]. When a cell arrives at an SE in any stage of the Shuffleout, its tag field is updated by a distance computing logic of the SE through re-computing the distance of the cell from the current SE to its destined SE.

A 2×4 SE requires two distance computing logics, each of which is complex and takes some 200 gates to implement [5]. Additionally, the complexity of these computing logics grows quickly as the fabric size increases. This tag re-computation is needed at every SE, unlike our proposed switching fabrics, whose packet tags are never recomputed during the course of delivery inside the fabrics.

III. Proposed Switching Fabrics

Scalable switching fabrics are introduced for high-performance routers operating at aggregate speeds of multi-terabits per second. They are necessary in distributed-router architecture [3] for providing connections among LC's, and in parallel-router architecture for providing connections between LC's and FE's (forwarding engines, which serve to look up IP addresses in a tree-like data structure of prefixes, called a *trie* [11], according to the longest prefix matching software algorithm implemented therein).

A. Proposed design

Our proposed design includes two types of basic building blocks, one for providing routing decisions and connecting external links to the fabrics, called routing units (RU's), and the other for interconnecting RU's based on a grid style. A design of size 256 is depicted in Fig. 1, where an RU is a small crossbar for connecting two external links (through its external ports) plus up to three *x*-directional connecting components (CC's) and up to three *y*-directional CC's (through its internal ports), with a CC being a multistage interconnect, like the Omega network [7], composed of 4×4 crossbars. Such a CC is known to be scalable with reasonably low hardware complexity, and it routes packets in a distributed manner according to the destinations of the packets. RU's are interconnected by CC's following a 2-D grid fashion, with N_x and N_y RU's along the

x and the y directions, respectively, for a fabric of size $2\cdot(N_x\times N_y) = N$. Here, we assume that N_x and N_y are multiples of 4, since CC's are composed of 4×4 crossbars. The proposed switching fabrics are dubbed GMR (i.e., grid-oriented, multistage-connected RU's) fabrics.

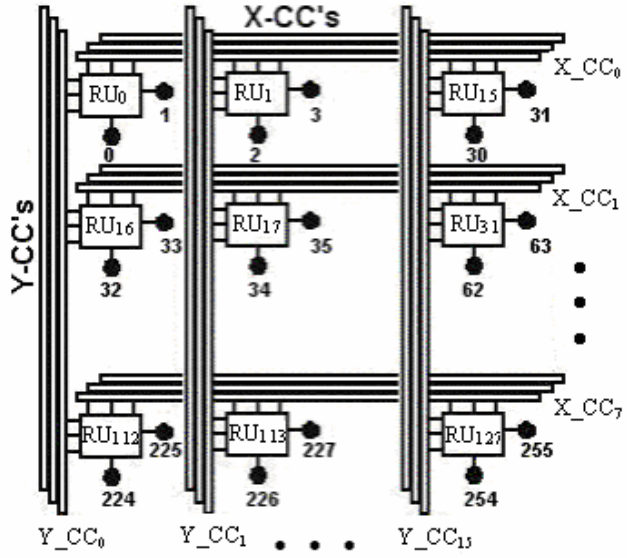

Figure 1. Proposed GMR of size 256.

Routing Units. An RU connects two external links via its external ports to the switching fabric. A GMR with size N contains a total of (N/2) RU's, which are numbered from RU_0 to $RU_{(N/2)-1}$, as shown in Fig. 1. The external ports in those RU's are labeled from 0 to N-1. Given an external port, say, numbered p ($0\le p\le N-1$), the RU to which the port is connected can be derived immediately by a shift register (to the right by one bit). Upon admitting a packet, an RU produces the routing tag of the packet based on its destination port address, say p, as follows. Given a size $N = 2\cdot(N_x\times N_y)$, the tag is a 2-tuple $<t_x, t_y>$, where t_x (or t_y) is the x (or y) coordinate of the RU to which the destination port is connected, namely, t_x equals the remainder of $(p/2)/N_x$ and t_y is the quotient of $(p/2)/N_x$. Computing t_x and t_y is done easily using a pair of shift registers, if N_x is a power of 2. Alternatively, one may pre-compute the tag of every port number and keeps the computed tags in a small on-chip SRAM for quick lookups (with an access time of about 1 *ns*), for an arbitrary N_x. Each RU has an ID register pair for holding its x and y coordinates. The ID register pair is loaded with proper values, depending on the RU position, every time when the router is powered up. If a packet is destined for an external port, which is connected to the same RU where it arrives, the packet is routed to its destination by the RU directly without going through any CC. Otherwise; the packet is delivered to its destination by taking either an x-directional CC, a y-directional CC, or both. Note that a packet takes at most one x-directional CC and one y-directional CC before reaching its destination. An RU handles two types of traffic, one is newly injected from its two external ports and the other is pass-through (transient) traffic, which was injected earlier at other RU's before being routed over an x-directional (or y-directional) CC to reach this RU. On receiving a packet, the RU checks its tag to see if the packet is destined for one of its two external ports (by simply comparing the packet tag with the contents of the RU's ID register pair). If the packet is not destined for the RU, it is routed forward over a CC according to the routing algorithm implemented in the RU (as detailed later). It should be noted that packets arriving at RU's are of fixed sizes for delivery to their destinations. A packet is sent by the RU to a CC, if needed, in one cycle and is moved in a cycle from one crossbar within a CC to another crossbar in the next stage. Each RU permits two links to terminate at its external ports (although this arrangement can be generalized to accommodate different link speeds, as explained later).

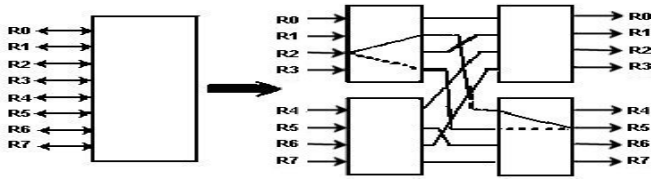

Figure 2. A CC of size 8 comprising 4×4 crossbars.

Connecting Components. Each connecting component (CC), whether an x-directional or a y-directional one, is composed of identical 4×4 crossbars. Routing in a CC is distributed, on the basis of two bits of the x (or y) coordinate of the destination tag of a packet, if it is x-directional (or y-directional). Fig. 2 depicts a y-directional CC, for the GMR of N = 256 shown in Fig. 1, and a path from R_2 to R_5 in the CC is highlighted for a packet with the y coordinate of its tag equal to 5 (i.e., 101_2). In this case, the first crossbar visited uses "×1" for its setting control, with "×" (a don't care) chosen as "0" and "1" being the leftmost bit of 101_2, while the second crossbar visited is set based on "01" (which are the next two bits of 101_2). Another path exists between R_2 and R_5, and the path (denoted by dashed line segments) is taken if the first crossbar visited chooses "×" to be "1" instead. Traffic from R_2 to R_5 is distributed uniformly over these two paths. For a CC with size 16 (or any power of 4, say 4^η), however, a tag coordinate employed for routing control consists of 4 (or 2η) bits and there is exactly one path between any communication pair.

A packet takes at most one x-directional CC and one y-directional CC before reaching its destined RU. The sequence of taking the two CC's, if needed, is decided by the routing algorithm implemented in the RU (at which the packet arrives). Note that an RU may connect up to three x-directional CC's (and also the same number of y-directional CC's), through its internal ports.

B. Routing algorithms in RU's

The routing algorithm implemented in each RU dictates the GMR behavior. It delivers packets to their destinations through

shortest paths without causing any deadlock. As only shortest paths are considered, routing in the GMR fabrics is always *minimal* [9]. It orders fabric resources (i.e., directions) and requires packets to use those resources in a strictly monotonic order, like the XY routing algorithm [12]. A packet is sent along an *x*-directional CC first before a *y*-directional CC (when both directions have to be taken) to avoid deadlocks. In this case, an RU makes its routing decision independent of traffic over the two directions, referred to as deterministic routing. When the packet is to be sent over an *x*-directional (or *y*-directional) CC, the RU then selects an arbitrary copy (among those multiple CC's, if existing) for delivery. Traffic is distributed over those multiple copies uniformly. This deterministic routing is called the *XY routing* algorithm subsequently.

If an RU connects two or three CC's along each direction, another way to avoid deadlocks is to designate two CC's (or one CC) in both dimensions as Layer A and the remaining as Layer B; a packet, when requiring to take both directions, has to go through a CC of Layer A (in either direction) first before a CC of Layer B, satisfying that resources (i.e., copies of CC's) are employed in a strictly monotonic order. Traffic over the CC's of Layer A is uniformly distributed, so is traffic over the CC's of Layer B. This deterministic routing again is minimal and deadlock-free, called the *Layered routing* algorithm. Performance is found to be slightly better under XY routing than under Layered routing based on our simulation results. Therefore, we will not discuss further the Layered routing algorithm.

C. Performance enhancements

The RU's and the CC's used for building GMR all adopt *output queuing*, where buffers are present only at the output ports. In general, output queuing is simpler than input queuing but often requires the building blocks (BB's) to run faster than the speed of lines (i.e., links) which connect BB's, because multiple packets may contend for an output and they need to be moved to the associated output queue in one cycle. If a BB (whether an RU or a CC) is *twice* as fast as the speed of its links, up to two contending packets can be sent to one output queue in a cycle. In this case, the BB is referred to have a *speedup* (ζ) of 2. Performance is enhanced with $\zeta = 2$ in comparison with $\zeta = 1$, and a larger ζ yields a higher performance figure (i.e., throughput) [3]. If RU's operate at a slow rate of 100 MHz (like the core of the six-port SGI Spider chip designed for distributed endpoint interconnects [6]), it is possible for RU's to have $\zeta_{RU} = 4$ or more. In practice, however, it will be shown by our simulation results in the next section that $\zeta = 3$ is enough to achieve desired performance levels. On the other hand, CC's are far simpler than RU's and may operate at a higher rate; say 200 MHz or 300 MHz. With the same design technology as that of RU's, CC's then can achieve a speedup (ζ_{CC}) of no more than 2.

The second enhancement overcomes a disadvantage resulting from deterministic routing, which requires packets to be routed in a specific resource order so that deadlocks are avoided.

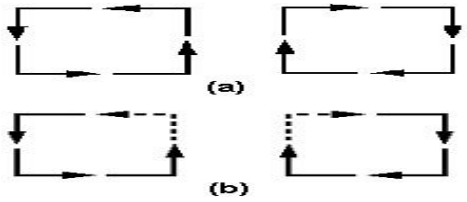

Figure 3. (a) Eight possible turns in a 2-D grid. (b) Six turns allowed by the North-Last policy.

By relaxing the strict order in utilizing resources during routing, performance of GMR can be improved by routing packets adaptively in accordance with the traffic situation in both directions. This *adaptive routing*, however, cannot guarantee deadlock-freeness unless a certain routing policy is enforced or additional resources are added. Here, an elegant routing policy, known as the turn model [9], is applied to ensure deadlock-freeness under adaptive routing. In this study, the North-Last policy is chosen and with such a policy, packets are allowed to travel in the north direction only at their last hops, as shown in Fig. 3. Under this North-Last adaptive routing, packets may be delivered along *x*-directional CC's without constraints, but they can be routed along y-directional CC's to the north (i.e., to lower port numbers) only at the last hop; they may be routed along *y*-directional CC's to the south (i.e., to higher port numbers) with no constraints. It is dubbed the *ANL* (Adaptive with the North-Last policy) routing algorithm, whose performance will be presented later on.

D. Other design alternatives

Each RU serves to terminate external links at its ports (like the role of a line card, LC) and to route packets based on the routing algorithm implemented therein. Like any LC in commercial routers (e.g., Cisco 12000 Series routers and the Juniper M160 backbone router), an RU admits links with various speeds. If an OC-192 link is connected, the RU may admit merely one such a high-speed link, with the other port left unused. If Ethernet links, or ATM links with 25 Mbps or with the OC-1 speed are connected, however, more than two links can terminate at an RU. In this case, the same RU can be used, but it is used in a reverse way, namely, up to three links may aggregate at one port (for a total of six slow links per RU), and it connects only one CC along each direction, following simple XY routing. This reverse use is possible because all RU ports are bi-directional, and control signals and the buffer of each port are identical in either use. Note that, if every RU in GMR is assumed to take exactly two port numbers (even though one of the two ports may be inactive), packet tag generation is simple and can be carried out by hardware at arrival RU's; otherwise, it has to resort to table lookups to get the routing tag (i.e., $<t_x, t_y>$) of a given destination address.

Our discussion so far has been on the fabric configurations where RU's are at the cross-points of 2-D grids, each for connecting up to three CC's along *x*- and *y*-directions. This type of configurations allows the fabrics to grow cost-effectively. For extremely large construction (say, $N > 8192 = 2 \cdot 64 \times 64$), a 3-D grid may be preferred, with RU's at its cross-points connecting CC's along three dimensions. A different RU is then required to connect multiple CC's in each dimension. The routing algorithm implemented in each RU either can be an extension of XY routing or ANL routing described above, or follows the deadlock-free routing procedure realized in the Hitachi SR2201 [15]. On the other hand, the same CC's can still be used for this 3-D grid configuration.

IV. Performance Evaluation and Comparison

Performance of GMR with various sizes is evaluated using simulation. A CC in GMR is composed of small 4×4 crossbars, which are simpler than an RU and may operate at a higher clock rate. An RU has two ports for external connections and six ports for up to three connections each along the *x*-direction and the *y*-direction. When compared to the SGI's Spider chip (which has six full-duplex ports developed for high-end networking applications [6]), an RU appears to have simpler routing because it routes packets among the two external ports and two groups of three ports each, with one group for one direction. In other words, an RU is based on a partially multiplexed 8×8 crossbar, whereas the Spider chip employs a fully multiplexed 6×6 crossbar. Each link of the Spider chip operates at a 200 MHz differential clock in each direction, sampled at both edges of every cycle to achieve the effective data rate of 400 Mbps. In this work, RU's are assumed to operate at a clock rate of 100 MHz (like the core of the Spider chip [6]). Each visit to an RU thus takes 10 *ns*, which is denoted as *one cycle time*. Let the ratio of the CC speed to the RU speed be represented by γ, then CC's are operating at the clock rate of $(100 \times \gamma)$ MHz. In our simulation, each external port generates a fixed-length packet following an independent Bernoulli process. Each packet can be moved from an RU to a 4×4 crossbar within a CC in one cycle time (of 10 *ns*), while it can be moved from one 4×4 crossbar to another within a CC in $(10/\gamma)$ *ns*. A queue of capacity 4 is associated with each output port (following output queuing implementation [3]).

The performance measures of interest include throughput and mean latency. Throughput indicates the number of packets delivered to a fabric output per cycle time, while mean latency reflects the average time for a packet to travel from its arrival port to its destined port. The drop rate of packets in GMR with size 256 is also gathered and compared with that of its Shuffleout counterpart [1], under heaviest possible traffic. Each data value presented is the result after 100,000 simulation clock cycles; this number has been observed to produce stable results for all evaluated cases.

The simulation results for GMR with $N = 32$ and $\zeta_{CC} = 2$ under XY routing are depicted in Fig. 4, where the mean latency is given in *ns*, Γ is the number of CC's connected to an RU along each direction, and ζ_{CC} is the speedup of constituent crossbars in CC's, representing the number of packets which can be sent to one output queue in one cycle inside the crossbars.

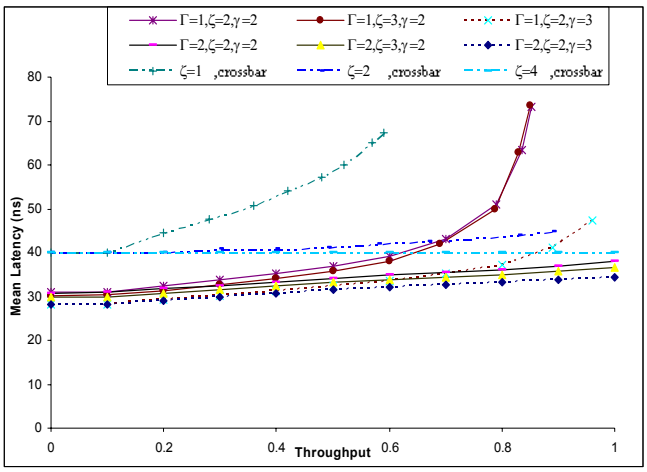

Figure 4. Mean latency versus throughput for N = 32 and $\zeta_{CC} = 2$ under XY routing.

In this figure, there are four solid curves for the cases of $\gamma = 2$, with two of them corresponding to $\zeta_{RU} = 2$ and the other two indicating $\zeta_{RU} = 3$ (where ζ_{RU} is the RU speedup referred by ζ in the figure). When throughput is low, say < 0.4, the four cases exhibit a similar mean latency. However, as throughput increases, the two cases with $\Gamma = 1$ start to experience fast growing mean latencies, signifying that those cases cannot deliver packets as swiftly if traffic in the switching fabric is not very light. Moreover, those two cases reach the maximum throughput of roughly 0.85, where mean latencies become excessive (regardless of the ζ_{RU} value). This is undesirable because the switching fabric then drops packets significantly when traffic is moderate or high. An acceptable switching fabric configuration drops few packets, if any, even under extremely heavy traffic, preferably exhibiting a *maximum throughput of 1.00* (or more precisely, a drop rate of less than 5×10^{-3}). For simple notation, we use AO (almost one) to represent 1.00 or better in throughput. The cases with $\Gamma = 2$ greatly extend their maximum throughputs, for different values of ζ_{RU} and γ, where the maximum throughput equals AO (i.e., the drop rate is then negligible). It is clear from Fig. 4 that $\Gamma = 2$ is sufficient for GMR with N = 32 under XY routing. The two cases of $\gamma = 3$ are shown by dotted curves in Fig. 4. It can be seen that the case of $\Gamma = 1$, that though the latency is higher, its maximum throughput approaches 0.96 due to a higher value of γ, with a higher latency. For the case of $\Gamma = 2$, the maximum throughput equals AO, and its mean latency remains small, even when throughput is AO. This is clearly the most desirable GMR configuration, among all the six cases

depicted in this figure. For comparison, the performance results of a state-of-the-art crossbar with size 32 under various speedups (i.e., ζ = 1, 2, 4) are simulated and shown by dashed curves in Fig. 4.

The compatible crossbar examined is assumed to have the cycle time of 40 *ns* (according to the data sheet of PMC-Sierra Inc.'s TT1 Chipset similar to those components employed for constructing Cisco high-end 12000 Series routers, due to its slow centralized scheduler, which operates at 25 MHz [10]). Such a state-of-the-art crossbar can reach throughput of AO when ζ equals 4, but its performance is clearly inferior to that of GMR with $\Gamma \geq 2$ for any value of ζ_{RU} and γ, as demonstrated in Fig. 4. Since GMR with a larger Γ (or ζ_{RU} or γ) yields better performance, namely, smaller mean latencies for any given throughput, GMR with $\Gamma \geq 2$, $\zeta_{RU} \geq 2$, and $\gamma \geq 2$ will outperform its crossbar counterpart as well.

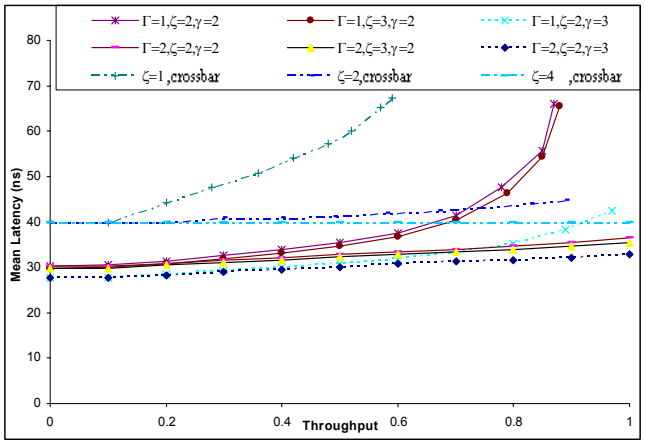

Figure 5. Mean latency versus throughput for $N = 32$ and $\zeta_{CC} = 2$ under ANL routing.

The simulation results for GMR with N = 32 and ζ_{CC} = 2 under ANL routing are illustrated in Fig. 5, where the same number of cases as those considered in Fig. 4 are included. From the six GMR curves, it is found that three cases (i.e, those with Γ = 2) are acceptable as their maximum throughput reaches AO. In addition, when a GMR curve in Fig. 4 is contrasted with its corresponding one in Fig. 5, it is observed that ANL routing yields slightly better performance than XY routing (namely, smaller mean latencies for given throughput values), as expected. This results directly from the routing flexibility offered by ANL routing, and the benefits become increasingly noticeable when traffic in GMR grows (and particularly so for throughput beyond 0.9). Again, GMR with Γ = 2 and for all values of ζ_{RU} and γ, outperforms its crossbar counterpart over the entire throughput range.

In Fig. 6, we depict the simulation results for a larger GMR configuration (i.e., N = 256), with ζ_{CC}, ζ_{RU}, and γ fixed to 2, 3, and 3, respectively, when Γ ranges from 1 to 3. The dotted curves denote XY routing, while the solid ones denote ANL routing. It can be found that, for the given parameter values, Γ = 1 is not acceptable for either routing since it leads to maximum throughput of around 0.7 only.

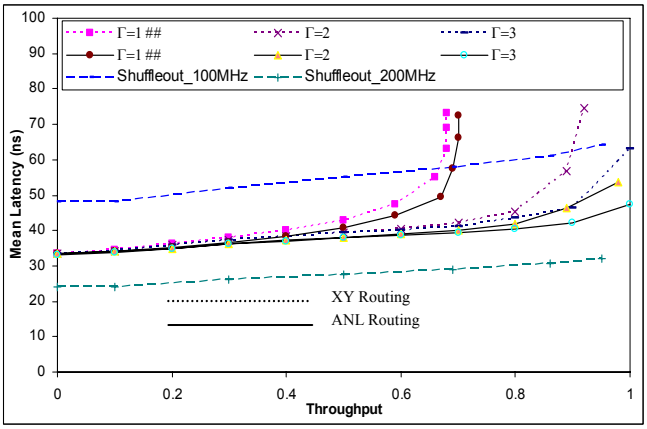

Figure 6. The impact of Γ on GMR performance for N = 256, $\zeta_{CC} = 2$, $\zeta_{RU} = 3$, $\gamma = 3$.

These results were obtained by randomly dropping a fraction of the packets whose latencies are excessive, to alleviate heavy congestion due to Γ=1.

On the other hand, Γ = 2 falls short of an AO throughput level only by a small amount, though for XY routing the delay is considerably high. Therefore, Γ has to rise to 3 under XY routing before GMR becomes acceptable. Under any given Γ, GMR gives rise to better performance with ANL routing than with XY routing, as expected. The simulation results of a compatible multistage-based switching fabric, called the Shuffleout [1], are included in Fig. 6 for comparison. The results are obtained following two assumptions: (1) the Shuffleout consists of ten (10) stages of 4×8 crossbars, with a packet therein taking 1 basic cycle (of 5 *ns* or 10 *ns* corresponding to the 200MHz curve or 100MHz curve) to move from one constituent crossbar to a neighboring crossbar, and (2) each output queue in the Shuffleout has a speedup of 10 (i.e., it may accept up to 10 packets per cycle in each queue) through its associated 10-to-1 multiplexor, which takes one cycle (of 10 *ns*) to pass through. The complexity of a Shuffleout SE is more than that of a CC and perhaps approaches the complexity of an RU. As a result, the Shuffleout is expected to operate at a rate between 100MHz and 200MHz. The two curves for Shuffleout are for the clock rates of 100 MHz and 200 MHz, respectively. From the standpoint of each Shuffleout's output queue (which is assumed to be able to receive upto 10 packets per cycle), however, the clock rate of the Shuffleout is more likely to be close to 100 MHz, since on-chip SRAM (Static RAM) has a typical access time of 1*ns* or more. It can be observed that the maximum throughput achieved by Shuffleout with 10 stages is only 0.96, which falls short of the maximum sustained throughput for our GMR with $\Gamma \geq 2$, under ANL routing, as shown in Fig. 6. GMR thus outperforms its Shuffleout counterpart, which fails to reach an AO throughput level and whose performance lies in

between the two curves shown. This is achieved even though the hardware complexity of Shuffleout is much higher (to be detailed in the next section).

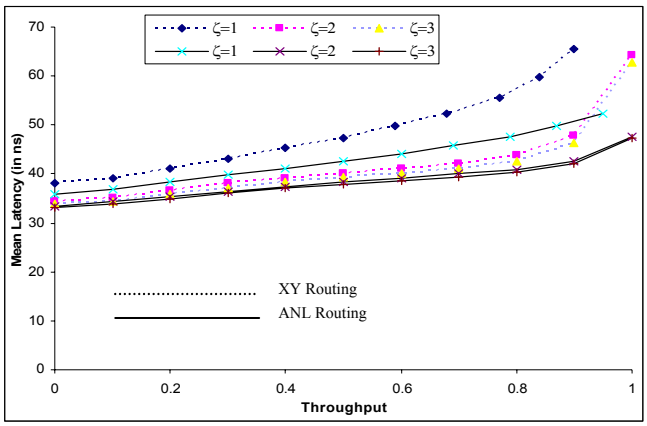

Figure 7. The impact of RU speedup (ζ) on GMR performance for $N = 256$, $\Gamma = 3$, $\zeta_{cc} = 2$, $\gamma = 3$.

Since $\Gamma = 3$ is sufficient for GMR to yield the maximum throughput of AO with reasonably low mean latencies, from this point onward, we shall limit our attention to the cases of $\Gamma = 3$ only. Fig. 7 shows the impact of RU speedups (denoted by ζ) on GMR performance, under XY routing (dotted curves) and ANL routing (solid curves), for $\zeta_{CC} = 2$ and $\gamma = 3$ with the fabric size of 256. It is clear from these curves that an RU speedup ≥ 2 does lead to considerable performance improvement as compared to a speedup = 1, in particular, when traffic is heavy, and that $\zeta_{RU} = 2$, for this configuration, is adequate to achieve the maximum throughput of AO, yielding acceptable performance levels. The results in this figure also signify that ANL routing delivers better performance than XY routing for any ζ_{RU}, as expected. Under XY or ANL routing, the performance gap between $\zeta_{RU} = 2$ and $\zeta_{RU} = 3$ is no longer pronounced even for extremely heavy traffic. This suggests that a preferred, cost-effective choice of ζ_{RU} is 2 under XY or ANL routing.

The impact of γ (i.e., the ratio of the CC speed to the RU speed) on GMR performance is illustrated in Fig. 8, where ζ_{RU} is chosen to be 3. Only under XY routing does $\gamma = 2$ fail to bring the maximum throughput to 1.0 though it reaches 0.98, but with a high latency. When γ rises to 3, both routing mechanisms exhibit the maximum throughput of AO, but ANL routing noticeably outperforms XY routing especially under extremely heavy traffic in terms of mean latency. As can be seen from the curves in this figure, ANL routing consistently performs better than XY routing for any given γ. In addition, for XY routing (or ANL routing), the preferred cost-effective choice of γ is 3 (or 2) since the performance gain between $\gamma = 2$ and $\gamma = 3$ is considerable (or insignificant) under heavy traffic.

We next compare the Shuffleout with our GMR in terms of the drop rate, defined as the percentage of packets being dropped under maximum possible input load of 1.0 (meaning that every fabric input produces one packet per RU cycle).

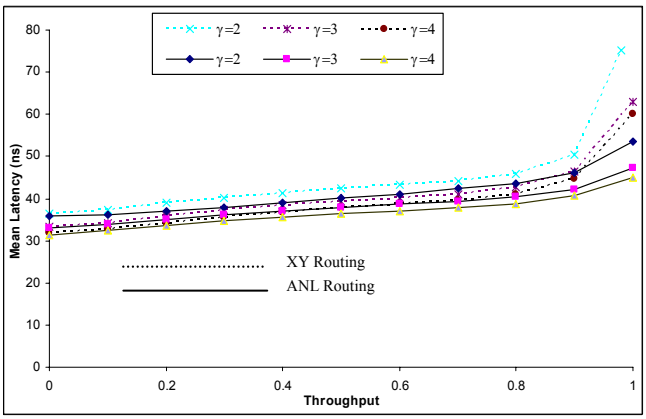

Figure 8. The impact of γ on GMR performance for $N = 256$, $\Gamma = 3$, $\zeta_{cc} = 2$, $\zeta_{RU} = 3$.

If a produced packet cannot be handled by the fabric at any point of time, it is dropped. A more efficient fabric exhibits a smaller drop rate. The drop rate as a function of ζ_{RU} for $N = 256$ is depicted in Fig. 9, where γ ranges from 2 to 4 under both XY routing and ANL routing.

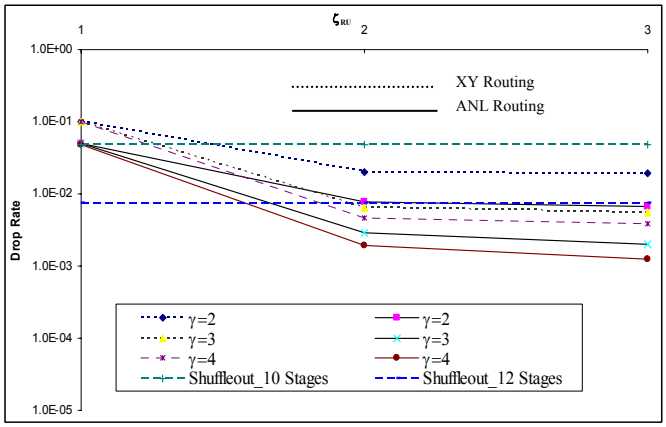

Figure 9. Drop rate versus ζ_{RU} for varying γ values under $N = 256$, $\Gamma = 3$, $\zeta_{CC} = 2$.

For any given γ, the drop rate reduces quickly as ζ_{RU} grows from 1 to 2, but there is a lesser reduction seen when ζ_{RU} increases from 2 to 3, under either routing mechanism. Similarly, the drop rate shrinks drastically as γ increases form 1 to 2 and a smaller reduction when γ increases from 2 to 3 under either routing, for a given ζ_{RU}. The drop rates of Shuffleout with 10 stages and 12 stages (of 4×8 crossbars) are included for comparison, where the Shuffleout results are obtained following the same two assumptions outlined earlier (for Fig. 6). As can be found in this figure, GMR under XY routing enjoys a smaller drop rate for $\zeta_{RU} \geq 2$ and $\gamma \geq 2$ than Shuffleout with 10 stages. When ANL routing is

applied, GMR with $\zeta_{RU} \geq 2$ and $\gamma \geq 2$ experiences a drop rate lighter than that of Shuffleout with 10 (or 12) stages. This is achieved despite the fact that our GMR possesses far lower hardware complexity than Shuffleout with 10 stages (and let alone a 12-staged Shuffleout), as detailed in the next section, and that optimistic assumptions are made in gathering the Shuffleout results. Apparently, GMR is superior to Shuffleout.

V. Complexity and Implementation Issues

The hardware complexity of GMR is examined and compared with a crossbar (of size 32) as well as a compatible Shuffleout (of size 256). We carry out the hardware analysis in this section for $\Gamma = 2$ having RU's with four internal ports each for connecting with two CC's along each direction. The Shuffleout was proposed for delivering ATM cells (each of 53 bytes) and is known to outperform many other ATM switch designs [1]. Like our proposed fabrics, the Shuffleout is scalable and follows distributed routing, although the procedure to route a cell at each constituent switching element (SE) is complicated, requiring re-computation of the distance from the SE to the cell's destination [5].

For GMR with size $N = 32$ and each RU connecting only one CC along each direction (i.e., $\Gamma = 1$), there are 4 CC's (each of a 4×4 crossbar) along one direction, giving rise to 8 CC's in total. If an RU connects two CC's in each direction, i.e., $\Gamma = 2$, GMR contains 16 CC's. In addition, it requires 16 RU's, irrespective of Γ. On the other hand, a compatible crossbar requires 64 4×4 crossbars to build, plus a centralized scheduler to dispatch packets existing in those 32 inputs. According to simulation results presented earlier, GMR of size 32 with $\Gamma = 2$, $\gamma = 3$, $\zeta_{CC} = 2$, $\zeta_{RU} = 2$ can outperform its crossbar counterpart under both XY routing and ANL routing for the whole throughput range. An RU employed for constructing GMR with $\Gamma = 2$ has complexity higher than that of a 4×4 crossbar, but it is simpler than a 6×6 crossbar because the four internal ports (for connecting CC's) constitute two groups, one for the *x*-direction and another for the *y*-direction, and any routing decision for these four ports is made only to the group level (rather than the individual port level). It implies that the RU is based on a partially multiplexed 6×6 crossbar (not a fully one), with its complexity estimated to be probably no more than *twice* as complex as a 4×4 crossbar. This renders the overall hardware complexity of GMR lower than that of its crossbar counterpart (which takes 64 crossbars of size 4×4 plus a centralized scheduler to build).

Next, we consider larger sized configurations, say, $N = 256$. Let GMR be organized as $N_x \times N_y = 16 \times 8$ and be compared with the Shuffleout, since a crossbar is unlikely to scale to this size. Our simulation results demonstrated previously indicate that GMR with $\Gamma = 3$ can outperform its Shuffleout counterpart. For $\Gamma = 3$ and $N_x = 16$, an RU connects three CC's along each direction, with an *x*-directional (or a *y*-directional) CC composed of 8 (or 4) 4×4 crossbars. There are 8*3 (or 16*3) CC's along the *x* (or *y*) direction, amounting to 3*(8*8 + 4*16) = 384 4×4 crossbars. Additionally, our fabric requires 128 RU's.

Consider the Shuffleout built from 4×8 crossbars, with 4 outlets for connecting the next stage and the other 4 outlets terminating at the output queues (see Fig. 1, where the SE is of 2×4 crossbars). Each SE for the Shuffleout contains 4 distance computing logics and 4 outlet identifiers, in addition to the regular 4×8 switching logics which are already more complicated than the 4×4 crossbars employed for constructing our CC's, and those computing logics are complicated each requiring some 200 gates [5]. The number of 4×8 crossbars needed per stage is 64, and it takes 10 (or 12) stages to yield the maximum throughput of 0.96 (or 0.99). If 10 stages are considered, it requires 640 4×8 crossbars in total. In addition, such a Shuffleout employs a queue for each output port and the queue is preceded by a 10-to-1 multiplexor (for terminating links from the 10 stages of SE's), signifying that it totally needs 256 queues (each of capacity to hold 10 packets or more) and 256 multiplexors**.** Obviously, GMR possesses far less hardware complexity than a compatible Shuffleout.

Besides its low hardware complexity, GMR has advantages from the implementation standpoint as well. In order to allow any sized construction, two types of chips are suggested for GMR. A Type C chip includes two 4×4 crossbars, requiring 16 data pins for 1-bit link width (or 128 pins for 8-bit link width implementation). A Type R chip contains two RU's, with 2*(6*2) = 24 data pins for 1-bit link width (or 192 pins for 8-bit link width), where every port is bi-directional and corresponds to two pins to support full-duplex. (This is because each RU employed has four internal ports and two external ports.) For $N = 4$ and $\Gamma = 1$, it takes one Type C chip and one Type R chip to build, as depicted in Fig. 10(a). Similarly, it requires six Type C chips and four Type R chips to construct a GMR with $N = 16$ and $\Gamma = 2$, as demonstrated in Fig. 10(b).

For $N = 32$ and $\Gamma = 2$ (able to outperform the crossbar counterpart), the chip count equals 16 (i.e., 8 Type C chips and 8 Type R chips). This compares favorably with switching fabrics in the Cisco 12000 Series routers realized using components similar to the PMC-Sierra Inc.'s TT1 Chipset [10], which includes the dataslice chip, the crossbar chip, scheduler chip, and the EPP (enhanced port processor) chip. Note that the proposed two chips accommodate the functions of the first three chips only. This chipset permits to construct switching fabrics only up to size 16 (or 32, if half-duplex), since it implements one single crossbar with a centralized scheduler. For a small fabric, the Cisco routers still uses the same scheduler chip designed for size 16, in contrast to GMR whose routing control is distributed among RU's so that fewer RU's are then involved. For GMR with *N*

= 256, it takes 64 Type C chips and 64 Type R chips when Γ equals 1. For Γ = 2, the Type C chip count becomes 128 while the Type R chip count remains unchanged.

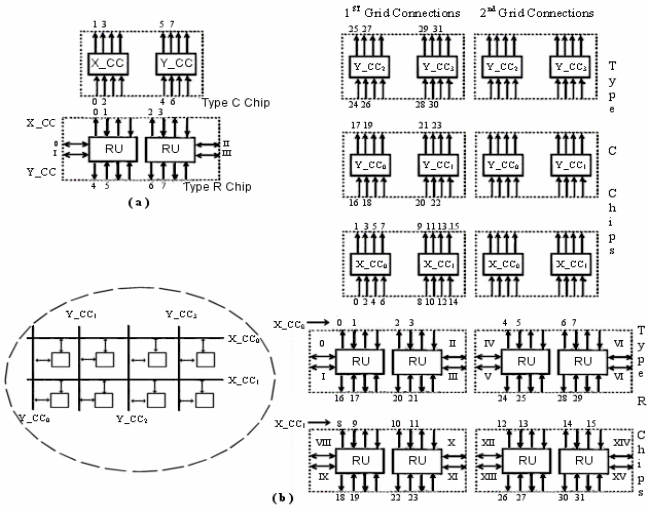

Figure 10. (a) GMR with $N = 4, \Gamma = 1$. (b) GMR with $N = 16$ (organized as $N_x \times N_y = 4 \times 2$), $\Gamma = 2$.

(Connected links between a Type C chip and a Type R chip are labeled with the same numbers. The connections in the 2nd grid are similar to those in the 1st grid and are thus omitted for clarity.)

VI. Conclusion

This paper has treated novel switching fabrics aiming at scalable routers with large numbers of line cards (LC's) to achieve the aggregate processing/forwarding rates of multi-terabits per second. The proposed switching fabrics consist of small RU's (routing units) interconnected by multistage-based connecting components (CC's) in accordance with a 2-D grid style, dubbed GMR (grid-oriented multistage-connected RU's) fabrics. Each RU is connected to one or multiple CC's along the *x* and the *y* directions, with packets routed in CC's following a distributed manner to embrace high scalability. GMR is evaluated by extensive simulation for various sizes and routing algorithms. In practice, an RU in GMR needs to connect only with three CC's along each direction for acceptable performance (with the maximum throughput reaching 1.00), for small speedups at RU's (say, $\zeta_{RU} = 3$) and at the crossbars of CC's (say, $\zeta_{CC} = 2$), with the ratio of the CC clock rate to the RU clock rate equal to 3. It is shown to have less hardware complexity and a lower cost (in terms of chip count) than its crossbar counterpart for small sized construction ($N \leq 32$) and yet exhibits superior performance. When the size (N) rises, GMR scales well in terms of cost and performance. For $N = 256$, GMR outperforms its Shuffleout counterpart, despite its far less hardware complexity. It requires only two types of chips for any sized construction with a low cost, ideally suitable for scalable routers.

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. on Communications*, vol. 42, pp. 2881-2889, Oct. 1994.

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

[3] S. Chuang, A. Goel, and N. McKeown, "Matching Output Queuing with a Combined Input Output Queued Switch," *IEEE J. Selected Areas in Communications*, pp. 1030-1039, June 1999.

[4] Cisco Systems, *Cisco 12016 Gigabit Switch Router, Data Sheet*. 2001, URL – http://www.cisco.com.

[5] 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.*, June 1991, pp. 27-34.

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

[7] D. Lawrie, "Access and Alignment of Data in an Array Processor," *IEEE Trans. on Computers*, vol. C-24, pp. 1145-1155, Dec. 1975.

[8] N. McKeown *et al.*, "The Tiny Tera: A Packet Switch Core," *IEEE Micro*, vol. 17, pp. 26-33, Jan./Feb. 1997.

[9] L. Ni and P. McKinley, "A Survey of Wormhole Routing Techniques in Direct Networks, *Computer*, pp. 62-76, Feb. 1993.

[10] PMC-Sierra Inc., *TT1 Chipset, Data Sheet*. 1999, URL – http://www.pmc-sierra.com.

[11] M. Ruiz-Sanchez, E. Biersack, and W. Dabbous, "Survey and Taxonomy of IP Address Lookup Algorithms," *IEEE Network*, vol. 15, pp. 8-23, Mar./Apr. 2001.

[12] C. Seitz *et al.*, "The Architecture and Programming of the Ametek Series 2010 Multicomputer," *Proc. 3rd Conf. on Hypercube Concurrent Computers and Applications*, Jan. 1988, pp. 33-36.

[13] W. Wang, K. Yang, "A Terabit Switch Fabric with Integrated High-Speed CMOS Trans-ceivers," *Proc. 8th Symp. on High-Performance Interconnects (Hot Interconnects 8), Aug.* 2000.

[14] T. Wolf and J. Turner, "Design Issues for High-Performance Active Routers," *IEEE J. Selected Areas in Communications*, vol. 19, no. 3, pp. 404-409, Mar. 2001.

[15] Y. Yasuda *et al.*, "Deadlock-free Fault-tolerant Routing in the Multi-dimensional Crossbar Network and Its Implementation for the Hitachi SR2201,"*Proc. 11th Int'l Parallel Processing Symp.*, April 1997 pp. 346-352.

[16] K. Yun, "A Terabit Multi-Service Switch," *IEEE Micro*, vol. 21, pp. 58-70, Jan./Feb. 2001.