Benes Switching Fabrics with $O(N)$-Complexity Internal Backpressure

Georgios Sapountzis† and Manolis Katevenis†
Institute of Computer Science - Foundation for Research and Technology - Hellas (FORTH)
ICS-FORTH, P.O. Box 1385, Heraklion, Crete, GR-711-10 Greece
http://archvlsi.ics.forth.gr/bpbenes/ - {sapunjis,katevenis}@ics.forth.gr

***Abstract*— Multistage buffered switching fabrics are the most efficient method for scaling packet switches to very large numbers of ports. The Benes network is the lowest-cost switching fabric known to yield operation free of internal blocking. Backpressure inside a switching fabric can limit the use of expensive off-chip buffer memory to just virtual-output queues (VOQ) in front of the input stage. This paper extends the known backpressure architectures to the Benes network. To achieve this, we had to successfully combine per-flow backpressure, multipath routing (inverse multiplexing), and cell resequencing. We present a flow merging scheme that is needed to bring the cost of backpressure down to $O(N)$ per switching element. We prove freedom from deadlock for a wide class of multipath cell distribution algorithms. Using a cell-time-accurate simulator, we verify operation free of internal blocking, we evaluate various cell distribution and resequencing methods, we compare performance to that of ideal output queueing and the iSLIP crossbar scheduling algorithm, and we show that the delay of well-behaved flows remains unaffected by the presence of congested traffic to oversubscribed output ports.**

Topics: HPSR Architectures, HPSR Analysis and simulation.

1 . INTRODUCTION

Switches, and the routers that use them, are the basic building blocks for constructing high-speed networks that employ point-to-point links. As the demand for network throughput keeps climbing, switches are needed with both faster ports and more ports. This paper concerns *switch scalability* when the *number of ports increases*. For low to modest numbers of ports –up to about 64– the crossbar is the switch topology of choice, owing to its simplicity and non-blocking operation. However, its cost grows with N^2, where N is the number of ports, which makes it very expensive for large N. Additionally, crossbar scheduling is a hard problem, and gets much harder with increasing N.

For switches with hundreds or thousands of ports, *multistage switching fabric* architectures are needed, whose cost growth rate is less than quadratic. The lowest-cost $N \times N$ network that is free of internal blocking is the *Benes* network [1], whose cost is $N \cdot 2logN$. The Benes network is *rearrangeably* non-blocking, that is, when each connection is routed through a single path, setting up new connections may require the re-routing of existing connections; however, using multi-path routing, this disadvantage can be eliminated. This paper concerns the Benes network.

If a multistage switching fabric contains no buffer storage, there must exist a mechanism to handle the cell routing conflicts that arise *(a)* in internal paths due to the routing algorithm, and *(b)* due to output conflicts. The former conflicts can be handled in a distributed manner ("self-routing fabrics") using Batcher sorting networks [2]. The latter conflicts – cells destined to the same output at the same time– must be avoided at the inputs or tolerated in the fabric. Avoidance at the inputs is equivalent to crossbar scheduling and requires global coordination, hence it is unrealistic for large fabrics. To tolerate output conflicts in the fabric, designers have used recirculation of cells [3] or multiple paths to each output buffer [4]. All of these mechanisms cost a lot in number of stages and paths per stage in the switching fabric: the fabric cost is $O(N \cdot log^2 N)$, and the constant in front of the actual cost is significant. In essence, these techniques spend (expensive) communication resources in order to economize on (inexpensive) storage resources, which is the wrong tradeoff in modern VLSI technology.

It is preferable for the switching fabric to contain internal buffer storage, in order to buffer conflicting cells until the conflict goes away. Such internal storage may be small enough to fit inside the switching-element chips, or it may be large enough to replace the buffer space typically found on the ingress line cards –usually hundreds of MBytes– hence requiring off-chip DRAM. In the former case, *backpressure* is used to prevent the small buffers from overflowing; effectively, the majority of the buffered cells are pushed back onto the ingress line cards, as in the usual case of virtual-output queues (VOQ) on the input side. Given that the ingress lines are much fewer than the intra-fabric links, this architecture results in significant cost savings when compared to off-chip DRAM for intra-fabric buffers [5]. Several commercial chip sets also use backpressure in the ingress-switch-egress connection chain [6] [7] [8]. This paper concerns the application of this advantageous *internal backpressure* architecture to the Benes network.

Contributions: In this paper, we extend the backpressure architecture from single-path to multi-path fabrics, and specifically to the Benes network. This extension is non-trivial. In order for the Benes fabric to operate free from internal blocking, the cells of each flow must be routed over multiple paths, and must afterwards be properly resequenced, as reviewed in section 2. In order for backpressure to operate free of head-of-

†The authors are also with the Dept. of Computer Science, University of Crete, P.O. Box 2208, Heraklion, Crete, GR-714-09 Greece

line-blocking effects, it must operate on a per-flow granularity, as reviewed in section 2. If these two requirements were combined in a naive way, $O(N^2)$ complexity would result for the switching elements in the middle stages of the Benes fabric. We show how to reduce this complexity down to $O(N)$, using appropriate flow merging techniques which minimally affect performance: see section 3. The resulting complexity of $O(N)$ is realistic for modern VLSI technology, because fabrics of size N in the order of a few thousand ports require on-chip buffer storage on the order of several thousand cells (several Mbits), which is feasible.

Multi-path cell distribution interacts with flow merging, and they both interact with the organization and placement of buffers; we show which organization is preferable, and we prove that it is deadlock-free (section 4). Finally, section 5 presents our simulation results, showing that: *(a)* non-blocking operation with full output utilization is indeed achieved; *(b)* the delay-versus-load characteristics of this switching fabric under bursty traffic are comparable within a factor of 1.5 to those of ideal output queueing; *(c)* delay to uncongested outputs is minimally affected by the presence of congestion (oversubscribed outputs) elsewhere in the network; and *(d)* delay is not very sensitive to the specific multi-path cell distribution method within the class of methods we consider.

To the best of our knowledge, this is the first time that the application of per-flow backpressure to the Benes switching fabric is studied. Also, we are not aware of other studies of backpressure with multi-path cell routing in general. Multi-path cell routing has been studied before, e.g. [9] [10] [11], but not with backpressure. Current switching fabrics use up to 3 stages [6] and employ algorithms that are optimized for the specific number of stages. We consider that our study is at least of theoretical importance and demonstrates the scalability of specific algorithms used in packet switching systems.

2 . The Benes Fabric

This section reviews the two foundations of our design: the Benes fabric, and internal backpressure in switches.

The Benes network [1] can be constructed recursively, using *inverse multiplexing* [12] [9], as shown in fig. 1. The $N \times N$ Benes network consists of two $\frac{N}{2} \times \frac{N}{2}$ Benes subnetworks, $\frac{N}{2}$ switching elements of size 2×2 connected at the inputs, and $\frac{N}{2}$ switches of size 2×2 connected to the outputs of the two subnetworks. The Benes network can also be constructed by placing two banyan networks back-to-back. The two banyans are called the *distribution* and the *routing* network, respectively [13], since the first distributes incoming traffic over the N links in the middle of the network and the second routes cells to the proper output link.

Non-blocking operation as above is based on (repeated) *inverse multiplexing* or *load distribution* in a balanced manner. To achieve balanced load distribution and still operate at the *cell* level a number of methods have been proposed: randomized [14], adaptive [9], per-flow round-robin cell distribution [15]. In all of these methods, cells of a given microflow are routed through either path, hence they may arrive out-of-order. For the switching fabric to preserve cell order within individual microflows, resequencers must exist at the points of path reconvergence [12] [10]. Resequencing is an important issue in our system, dealt with in sections 3 and 4.

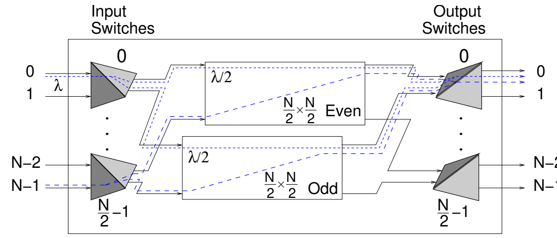

Fig. 1. Recursive construction of an $N \times N$ Benes network.

Switches with multistage buffering typically use *backpressure* feedback control between these stages, to avoid overflow of downstream buffers and to control individual flow rates when multiple flows merge into oversubscribed resources, thus enforcing quality-of-service (QoS) guarantees. We assume *credit-based* backpressure. Backpressure signals may refer to individual (micro) flows, or to flow aggregates, or indiscriminately to all traffic passing through a link. Indiscriminate backpressure leads to very poor QoS, because a single oversubscribed flow may stop the service to all other flows with which it shares a link or a buffer (this is analogous to head-of-line (HOL) blocking). Thus, *per-flow* or *virtual-channel* or *multilane* backpressure is needed. The number and definition of “flows” is a crucial parameter and affects cost – amount of state and granularity of feedback information– and QoS –degree of isolation among competing flows. This paper is concerned with full-fledged per-flow backpressure, which ensures that even if all output ports but one are oversubscribed, traffic going to that one non-congested output will still enjoy delays comparable to those of an ideal output-queued switch. We obtain such strong QoS guarantees at a cost not worse than $O(N)$ per switching element, which is realistic for modern VLSI technology.

The model assumed in this paper deals exclusively with the flow control *inside* the Benes fabric, independent of the type of flow control employed outside the fabric, in the overall network. Internal backpressure operates on flow aggregates that consist of all network-wide (micro-)flows that share a common path (and priority level) within the switching fabric. Thus, in the rest of this paper, for an $N \times N$ fabric with pl priority levels, we only consider the $N^2 \times (pl)$ flows defined, each, by one specific fabric input port, *i*, one specific fabric output port, *j*, and one specific priority level.

3 . Flow Merging and Queue Organization

In the Benes fabric the traffic of every flow is distributed and sent over both “even” and “odd” subnetworks in fig. 1; consequently, all subnetworks, no matter how small, down to the individual switching elements in the core of the fabric, are traversed by N^2 flows (per priority level). In this section,

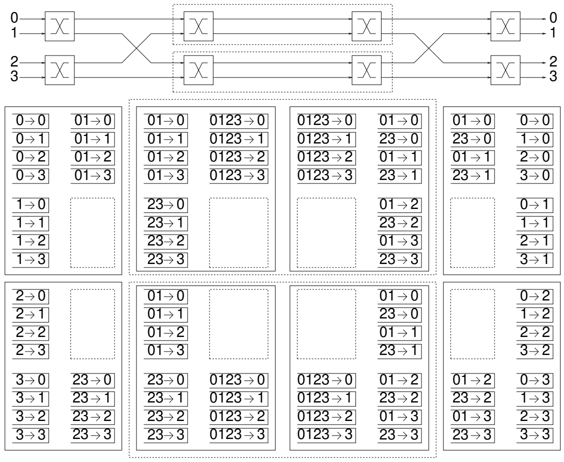

Fig. 2. A 4×4 fabric and the flow groups at the inputs and outputs of the switching elements for the per-output flow merging case.

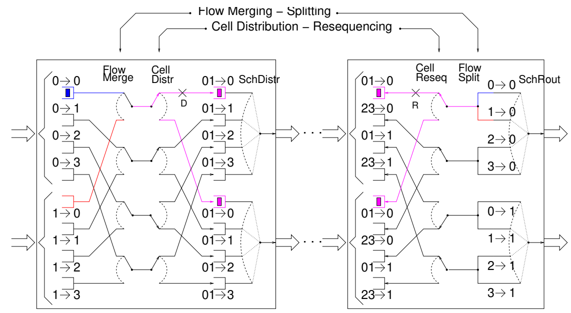

Fig. 3. Logical buffer organization of a distribution and the corresponding routing switching element.

we present a flow merging scheme that reduces the $O(N^2)$ backpressure cost (per switching element) down to $O(N)$. Next, we describe the queues and the functionality inside the distribution and routing switching elements.

In order to reduce the number of flows, we used *per-output* merging of the flows destined to the same output port of the fabric. Fig. 2 shows the *flow groups* that internal backpressure must operate on; "01 $\rightarrow$ 0" denotes the merging of flows $0 \rightarrow 0$ and $1 \rightarrow 0$, and "0123 $\rightarrow$ 0" is the merging of flow groups $01 \rightarrow 0$ and $23 \rightarrow 0$. This example uses 2×2 switching elements. Each switching element of the distribution network (left half of the Benes fabric) merges, one-by-one, the N flow groups entering through one of its inputs with the N flow groups entering through the other, and produces N merged flow groups; the merging factor is two-to-one. These switching elements also distribute the cells to both of their outputs, so the N merged flow groups appear on each of these outputs; fig. 2 shows one of these copies in full detail, and uses an empty box for the other. Hence, all links carry precisely N flow groups. (The two central stages of the fabric are shown separate for conceptual reasons, only; in reality, they are implemented as a single stage.)

In the routing network (right half of the Benes fabric), cells that had been distributed to the even and odd subnetworks must be resequenced. Resequencing, in output switches, must be performed separately for each flow in a merged flow group. The reason is that merged flow groups carry cells that were distributed at different input switches, independently of each other, before the merge points. Hence, merged flow groups from different inputs to a same output, must be split again in order for resequencing to work correctly.

Splitting of flow groups and cell resequencing can be performed progressively, per-stage, or cumulatively, in the very last stage of the fabric. In the latter case, we need not split flows within the routing banyan, thus, there would be $\frac{N}{2}$, ..., 2, 1 flows passing through the switching elements in the $\log_2 N$ stages of the routing banyan, respectively. However, each resequencer at the output ports of the fabric would then require N resequence buffers, one for each of the N (per-input) flows leading to that output, each of size $O(N)$. There is no reason to accumulate so much complexity in the last stage of the fabric, so we prefer the former solution –progressive flow group splitting and cell resequencing.

In conclusion, per-output flow merging with per-stage resequencing is much simpler to implement and has a uniform implementation cost of $O(N)$ per switching element, across all stages of the switching fabric, so we use this architecture in the rest of the paper.

Figure 3 shows the preferred logical buffer organization of the distribution and routing switching elements, along with the active components needed. We follow the flow merging and cell resequencing architecture that was chosen above. The flows from inputs 0 and 1 to four different fabric outputs are shown in the left (distribution) switching element, along with the flows to outputs 0 and 1 from four different fabric inputs in the right (routing) switching element. The FIFO's shown are *logical* queues, containing *references* to cells; the actual cells do not move inside the switching element.

4 . FREEDOM FROM DEADLOCK

The interleaving of multiple stages of cell resequencing and flow splitting combined with backpressure, has the potential danger of deadlock: a resequencer may be waiting for cells from a given path, while the splitter in the previous stage may be delivering cells in the wrong queue. We have shown that for a wide and interesting class of cells distribution methods, no deadlock situation can arise. In [16, section 3.3], we describe the potential deadlock situation and derive sufficient conditions for deadlock-free operation of the per-stage resequencers.

5 . SIMULATION RESULTS

A simulation model operating at the granularity of cell times was developed in order to verify the design and evaluate its performance under various traffic patterns and for various switch sizes, and in order to evaluate cell distribution and resequencing methods. In the simulation model, the cell-credit round-trip time is 1 cell time, and the buffer shown in fig. 3 has a size of 1 cell for the distribution switching elements, and 2 or 3 for the routing switching elements, depending on the cell distribution method.

We simulated the switch under smooth, bursty, and hotspot traffic. Smooth traffic consisted of Bernoulli arrivals with uniformly distributed destinations. The reported results use bursty/12 traffic, where the mean burst size is 12 cells; this is close to one of the modes of IP traffic size distribution (assuming 48-byte cell payload). Under hotspot traffic, each destination belonging to a designated set of "hot spots" receives (smooth or bursty) traffic at 100% collective load, uniformly from all sources; the rest of the destinations receive smooth or bursty traffic as above. The reported results use hotspot/4 traffic, where the four hotspots are ports 0, 1, 2, and 3. The delay reported is the average over all cells of the cell's exit time, minus the cell's birth time, *minus the fabric length* (number of stages) plus one; by subtracting the fabric length from the actual delay, we report the sum of all queueing delays for the cells plus one. In all of the reported results, the duration of the simulation is 200,000 cell times and collection of statistics starts after the first 40,000 cell times. We use 95% confidence intervals of 5%, except for one case where it was 7.1%.

To check for lack of internal blocking, we also simulated the 64×64 fabric under the following artificial load. In each and every cell time, a randomly-selected full permutation was presented to the input of the switch; that is, all inputs were continuously loaded at precisely 100%, while the overall load presented to the fabric was *feasible*, during each and every cell time. After one million simulation cell times, there were virtually no cells queued at the inputs: most of the VOQ's were empty, while a few others contained 1 or 2 cells each.

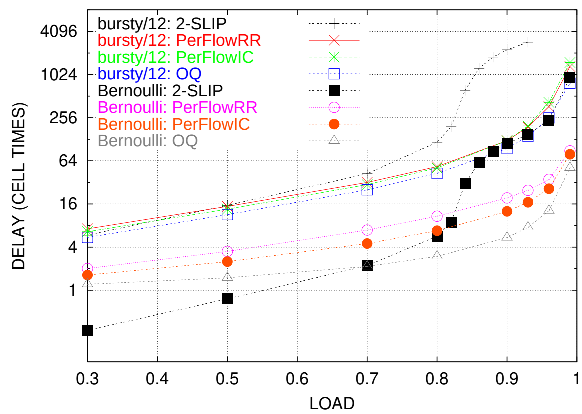

Fig. 4. Delay versus load for uniform destinations; 64×64 fabric made of 4×4 elements; upper curves: bursty/12 traffic; lower curves: Bernoulli traffic; ideal output queueing (OQ) also shown for comparison.

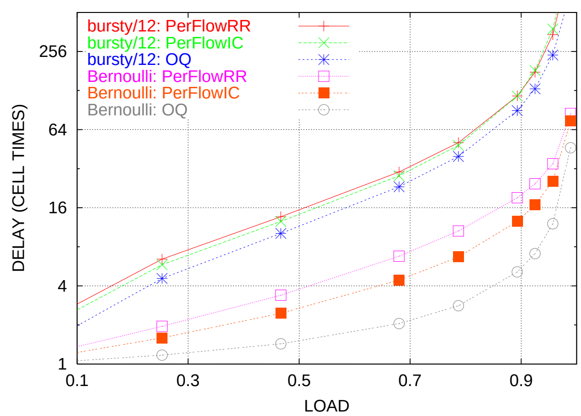

Fig. 5. Delay of non-hotspot destinations in the presence of hotspot/4 traffic; horizontal axis is the load to non-hotspot outputs; other parameters as in fig. 4.

5.1. Cell Distribution Methods

We experimented with two cell distribution methods, called *PerFlowRR* and *PerFlowIC*, on a 64×64 Benes fabric made of 4×4 switching elements. PerFlowRR is per-flow round-robin cell distribution, where the per-flow distribution pointers are randomly initialized. PerFlowIC (standing for per-flow imbalance count) chooses the port for forwarding the next cell as follows: among the set of ports that have received the least number of cells of this flow up to now, choose the port that currently has the least number of *ready cells*; ready cells are the cells (of any flow group) that are queued at this port and that have an available downstream credit. Both methods have a maximum per-flow *imbalance* of 1 – at any time, the total numbers of cells belonging to some flow that have been forwarded through any two paths available to that flow differs by at most 1 – and, in the long run, send the same number of cells in each path; PerFlowIC, though, is more flexible every time the imbalance count returns to 0. The results are shown in fig. 4, for uniformly destined traffic, and in fig. 5, for traffic in the presence of hot spots.

Under smooth (Bernoulli) traffic, the cell distribution method does make some difference: PerFlowIC yields 30% to 60% lower delay when compared to PerFlowRR. The difference is more pronounced for medium loads, and less pronounced for light or heavy loads. The presence or absence of hot-spot traffic does not affect this aspect of the results. Under *bursty* traffic, though, the cell distribution method makes virtually *no difference*. This must be due to the large number of back-to-back cells in the same flow: in this case, PerFlowIC becomes similar to PerFlowRR not only in the long but also in the short term.

5.2. Comparison with OQ and iSLIP

Figures 5 and 4 also show, for comparison, the delay of the ideal output-queued (OQ) switch under each traffic load; in every triplet of curves, output queueing is the lower of the three curves. We see that, under bursty traffic, the Benes fabric has only 20% to 60% worse delay when compared to ideal output queueing. Under smooth traffic, the switching fabric's delay is longer by a factor of 1.6 to 4, the difference being less pronounced for light load and more pronounced around 80% load. We also performed simulations for the two cell distribution methods under bursty/32 arrivals and either uniform or hotspot/4 destinations. Compared to ideal output queueing, the average delay was 10% to 60% higher for

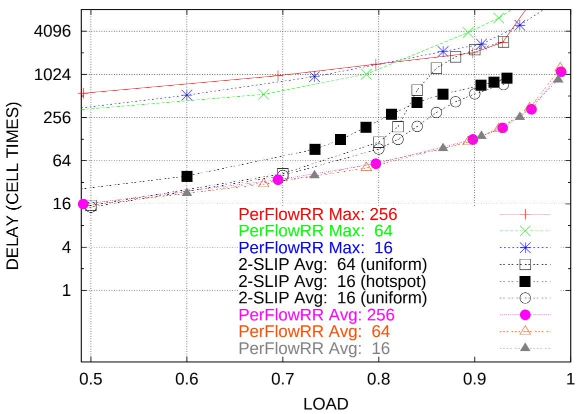

Fig. 6. Performance for various fabric sizes, 16×16 to 256×256: average delay and maximum delay versus load, under bursty traffic in the presence of hot spots. The results are for the PerFlowRR cell distribution method.

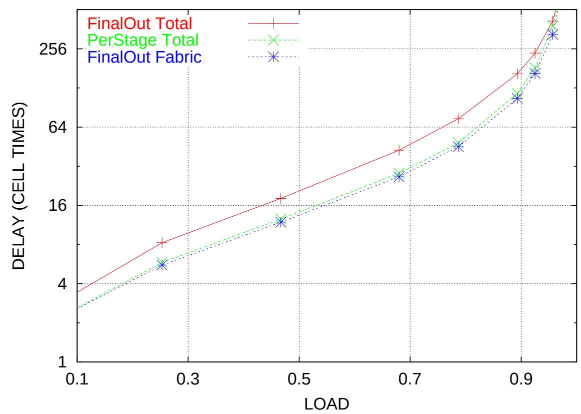

Fig. 7. Average delay under different resequencing methods; bursty traffic in the presence of hot spots. The results are for the PerFlowIC cell distribution method.

uniform destinations, and 15% to 85% higher for hotspot/4 destinations.

By comparing the delays in fig. 5 to those in fig. 4, we notice that they are almost identical, which shows that non-hotspot traffic stays virtually *unaffected* by the presence of hot spots in the network, thus proving the *excellent QoS properties* of this switch. Not shown in the plots is the throughput (utilization) of the hotspot destinations (remember that the load offered to them is 100%). Under smooth traffic this output utilization was consistently over 99%; under bursty traffic, it ranged from 92% to 98%.

Lastly, we compare the performance of the Benes fabric with that of a crossbar with VOQ's and the 2-SLIP crossbar scheduling algorithm [17] [1]. We see that, for loads under 70%, the delay for 2-SLIP is small, compareable to the delay through the Benes fabric. As the load gets higher, around 80%, the delay for 2-SLIP increases considerably, and for bursty traffic it is 14 to 18 times worse than the delay through the Benes fabric.

5.3. Fabric Size Dependence of Performance

One of the advantages of the proposed architecture is that it can scale to very large sizes. It is important for the performance of the fabric not to degrade with increasing size. We experimented with fabrics of up to 256 ports. We used the more "interesting" of the previous traffic patterns, bursty/12 arrivals with hotspot/4 destinations. The results are plotted in fig. 6, and they show that maximum cell delay generally increases with increasing fabric size, roughly by about 25% to 75% when the fabric size quadruples. However, *average* cell delay remains virtually *unaffected* by fabric size. We also present results for switches using the 2-SLIP crossbar scheduling algorithm. We see that in this case, for loads under 70%, average delay remains unaffected with increasing fabric size, but for higher loads it gets approximately 4 times worse with increasing fabric size.

[1]For the performance simulations for the 2-SLIP algorithm, we used the SIM simulator from Stanford University. The model for bursty traffic we used does not support loads over $\frac{b}{b+1}$, where b is the average burst size, thus, we present results for average loads up to 0.923%.

5.4. Alternative Cell Resequencing Methods

As discussed in section 3, cell resequencing can be performed progressively, "PerStage", or cumulatively, in the very last stage of the fabric ("FinalOut"). From the point of view of implementation, per-stage resequencing is simpler and less expensive than FinalOut, but the question regarding performance remains: it appears that FinalOut lets cells go faster through the routing network, and thus may lead to lower delays. In reality, things are the other way around!

Figure 7 shows the average delay under the two resequencing methods; input traffic is bursty/12 and hotspot/4, as in section 5.3. For the "FinalOut" method, we show separately the delay for the cells to get through the fabric, without yet being resequenced ("FinalOut Fabric"), and separately their total delay, including the resequencing process in the very last stage of the fabric ("FinalOut Total"). Interestingly, although cells do indeed get a bit faster through the fabric, as compared to the case where per-stage resequencing delays them in the routing network, when the delay of FinalOut resequencing is added, the overall delay of FinalOut is worse.

We see that letting some cells get quickly through the fabric, ahead of their order, without per-stage resequencing, appears to consume such fabric resources that, overall, it harms other cells more than it benefits the early-out cells. We conclude that *per-stage resequencing is strictly better* than cumulative resequencing in the very last stage of the fabric, both from the point of view of implementation cost and complexity as well as from the point of view of performance.

6. Conclusions

We showed how to efficiently scale packet switches to very large numbers of ports, while maintaining non-blocking operation and high quality of service. This can be done by extending the known per-flow backpressure architecture so as to make it applicable to the Benes fabric with multipath routing and cell resequencing. To the best of our knowledge, this is the first time that this combination of architectures is studied. In order to keep the cost manageable, we used an appropriate flow merging scheme that keeps the cost of backpressure down to $O(N)$ per switching element. Using a cell-time-accurate simulator, *(a)* we showed that per-stage resequencing is preferable; *(b)* we found that cell distribution based on imbalance leads to lower delays than round-robin distribution, but under bursty traffic this difference becomes negligible; *(c)* we noticed that delay under bursty traffic is only 20 to 60 % higher than ideal output queueing; *(d)* we showed that average delay is lower than for 2-SLIP, the difference being more pronounced for loads over 80%; and *(e)* we showed that the delay of well-behaved flows remains unaffected by the presence of congested traffic to oversubscribed output ports, thus proving the excellent quality of service properties of the system.

References

[1] V. Benes, "Optimal Rearrangeable Multistage Connecting Networks," *Bell Systems Technical Journal*, vol. 43, no. 7, pp. 1641–1656, July 1964.

[2] K. Batcher, "Sorting Networks and their Applications," in *AFIPS Proc. 1968 Spring Joint Computer Conf.*, vol. 32, 1968, pp. 307–314.

[3] A. Huang and S. Knauer, "Starlite: A Wideband Digital Switch," in *Proc. IEEE GLOBECOM '84 Conf., Atlanta GA USA*, Dec. 1984, pp. 121–125.

[4] J. Giacopelli, J. Hickey, W. Marcus, W. Sincoskie, and M. Littlewood, "Sunshine: a High Performance Self-Routing Broadband Packet Switch Architecture," *IEEE-JSAC*, vol. 9, no. 8, pp. 1289–1298, Oct. 1991.

[5] G. Kornaros, D. Pnevmatikatos, P. Vatsolaki, G. Kalokerinos, C. Xanthaki, D. Mavroidis, D. Serpanos, and M. Katevenis, "ATLAS I: Implementing a Single-Chip ATM Switch with Backpressure," *IEEE Micro*, vol. 19, no. 1, pp. 30–41, Jan. 1999. [Online]. Available: http://archvlsi.ics.forth.gr/atlasI/hoti98/

[6] F. Chiussi, J. Kneuer, and V. Kumar, "Low-Cost Scalable Switching Solutions for Broadband Networking: The ATLANTA Architecture and Chip Set," *IEEE Communications Magazine*, vol. 35, no. 12, pp. 44–53, Dec. 1997.

[7] "IBM PowerPRS Q-64G Packet Routing Switch Datasheet," Dec. 2001. [Online]. Available: http://www.ibm.com/chips/techlib/-techlib.nsf/products/PowerPRS_Q-64G_Packet_Routing_Switch

[8] "ETT1 Chip Set Datasheet," Mar. 2002. [Online]. Available: http://www.pmc-sierra.com/products/details/pm9312/index.html

[9] F. Chiussi, D. Khotimsky, and S. Krishnan, "Generalized Inverse Multiplexing for Switched ATM Connections," in *Proc. IEEE GLOBECOM Conf., Australia*, Nov. 1998, pp. 3134–3140. [Online]. Available: http://www.bell-labs.com/org/113480/Papers/fabio-globecom98B.ps

[10] D. Khotimsky, "A Packet Resequencing Protocol for Fault-tolerant Multipath Transmission with Non-Uniform Traffic Splitting," in *Proc. IEEE GLOBECOM Conf., Brasil*, Dec. 1999, pp. 1283–1289. [Online]. Available: http://www.bell-labs.com/org/113480/Papers/dkh-globecom99.ps

[11] S. Iyer, A. A. Awadallah, and N. McKeown, "Analysis of a Packet Switch with Memories Running Slower than the Line-Rate," in *IEEE INFOCOM*, Mar. 2000. [Online]. Available: http://klamath.stanford.edu/˜sundaes/Papers/infocom2000.pdf

[12] J. Duncanson, "Inverse Multiplexing," *IEEE Communications Magazine*, vol. 32, no. 4, pp. 34–41, Apr. 1994.

[13] J. Turner, "Design of a Broadcast Packet Switching Network," *IEEE Transactions on Communications*, vol. 36, no. 6, pp. 734–743, June 1988.

[14] L. G. Valiant and G. J. Brebner, "Universal Schemes for Parallel Communication," in *ACM STOC*, 1981, pp. 263–277.

[15] S. Iyer and N. McKeown, "Making Parallel Packet Switches Practical," in *IEEE INFOCOM*, Mar. 2001. [Online]. Available: http://klamath.stanford.edu/˜sundaes/Papers/infocom2001.pdf

[16] G. Sapountzis, "Benes Switching Fabrics with $O(N)$-Complexity Internal Backpressure," Master's thesis, University of Crete, Nov. 2002. [Online]. Available: ftp://ftp.ics.forth.gr/tech-reports/2002/2002.TR316.Bennes_-Switching_Fabrics_Complexity_Internal_Backpressure.pdf.gz

[17] N. McKeown, "iSLIP: A Scheduling Algorithm for Input-Queued Switches," *IEEE/ACM Transactions on Networking*, vol. 7, no. 2, Apr. 1999. [Online]. Available: http://tiny-tera.stanford.edu/˜nickm/papers/-ToN_April_99.pdf