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

Georgios Sapountzis

Master of Science Thesis

Department of Computer Science
University of Crete

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, also, 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 evaluated various cell distribution and resequencing methods, we found that delay under bursty traffic is only 25 to 50 percent higher than ideal output queueing, and we showed that the delay of well-behaved flows remains unaffected by the presence of congested traffic to oversubscribed output ports. Using simplified models of the Benes fabric, we show that cell distribution does not result in throughput limitations, and identify the points of the fabric where contention is resolved.

Thesis Advisor:
Manolis Katevenis
Professor, Department of Computer Science, University of Crete

Acknowledgements

First of all, I would to thank my advisor Manolis Katevenis for his patience and guidance throughout this work. I would also like to thank the members of the packet switch architecture group, N. Chrysos, V. Siris and P. Fragopoulou for valuable discussions and feedback and ICS-FORTH for providing infrastructure and financial support. Lastly, I would like to thank my friends and family for their support.

Contents

List of Figures

List of Tables

Chapter 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. Researchers have been looking at such scalable fabric topologies since the days of electromechanical telephony [1]. The banyan network [2] features a low cost, $N \cdot logN$ and a rich set of paths. Although it can support full egress link utilization under uniformly destined traffic, as well as a number of other specific traffic patterns, it does suffer from *internal blocking:* not all *feasible* rates $\lambda_{i,j}$ (see section 2.1.1) can be routed through it. The lowest-cost $N \times N$ network that is free of internal blocking is the *Benes* network [3], 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: see section 2.1.1. 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 [4]. 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 [5] or multiple paths to each output buffer [6]. All of these mechanisms cost a lot in number of stages and paths per stage in the switching fabric: the fabric cost is $O(N \cdot log^2N)$, 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 the off-chip DRAM case for intra-fabric buffers, as shown by the ATLAS I switch evaluation [7]. This paper concerns the application of this advantageous *internal backpressure* architecture to the Benes network –the lowest cost scalable switching fabric.

In this paper, we extend the backpressure architecture from single-path fabrics (like banyans) to multi-path topologies, and specifically to the Benes network. This extension is non-trivial. In order for the Benes fabric to operate free of internal blocking, the cells of each flow must be routed over multiple paths, and must afterwards be properly resequenced, as reviewed in section 2.1.1. In order for backpressure to operate free of head-of-line-blocking effects, it must operate on a per-flow granularity, as reviewed in section 2.1.2. 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 2.2.1. 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 3.3). Section 4 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 (over-subscribed 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. Finally, using simplified models of the Benes fabric, we show that cell distribution does not result in throughput limitations, and identify the points of the fabric where contention is resolved.

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. [8] [9] [10], but not with backpressure.

Chapter 2

Generic Architecture

2.1 The Benes Fabric

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

2.1.1 Non-blocking Operation

The Benes network [3] can be constructed recursively, using *inverse multiplexing* [11] [8], as shown in fig. 2.1. The $N \times N$ Benes network consists of two $\frac{N}{2} \times \frac{N}{2}$ Benes subnetworks, $\frac{N}{2}$ switches of size 2×2 connected to the inputs of the two subnetworks, called the input switches in this paper, and $\frac{N}{2}$ switches of size 2×2 connected to the outputs of the two subnetworks, called the output switches, here.

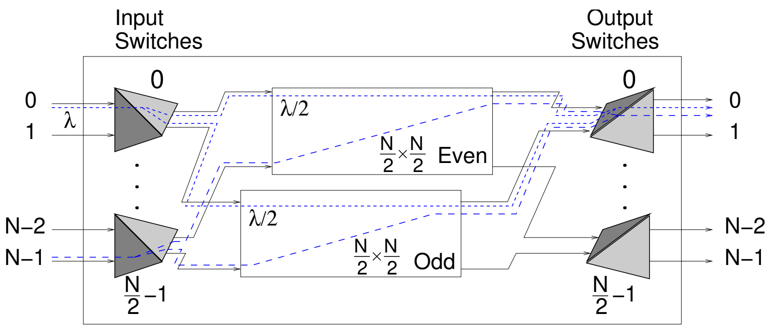

Figure 2.1: *Recursive construction of an $N \times N$ Benes network.*

Let $\lambda_{i,j}$ denote the traffic entering the network from input i and destined to output j. In order for the $N \times N$ network to be non-blocking, the 2×2 switch connected to input i must equally distribute $\lambda_{i,j}$ among its two outputs. The output switch that feeds output j receives $\frac{1}{2}\lambda_{i,j}$ on each of its inputs, reconstructs $\lambda_{i,j}$ and routes it to the appropriate output. Freedom from internal blocking results as follows. For any set of *feasible* rates $\lambda_{i,j}$ entering the $N \times N$ network (i.e. $\sum_{j=0}^{N-1} \lambda_{i,j} \leq 1, \forall i$) and leaving the $N \times N$

network (i.e. $\sum_{i=0}^{N-1} \lambda_{i,j} \le 1,\ \forall j$), the rates entering and leaving each $\frac{N}{2} \times \frac{N}{2}$ subnetwork will also be feasible. Specifically, input k of either subnetwork will be receiving $\sum_{j=0}^{N-1} \frac{1}{2}\lambda_{2k,j} + \sum_{j=0}^{N-1} \frac{1}{2}\lambda_{2k+1,j}$ which is $\le \frac{1}{2} + \frac{1}{2} = 1$ because of the above feasibility of the overall traffic. Symmetrically, the load of output m of either subnetwork will be $\sum_{i=0}^{N-1} \frac{1}{2}\lambda_{i,2m} + \sum_{i=0}^{N-1} \frac{1}{2}\lambda_{i,2m+1} \le \frac{1}{2} + \frac{1}{2} = 1$. Assuming that each subnetwork is internally non-blocking, i.e. can route any such feasible traffic, it follows by recursion that the overall $N \times N$ network will also be internally non-blocking.

Unrolling the recursion in fig. 2.1, for N = 8, results in the topology shown in fig. 2.2. Traffic $\lambda_{i,j}$ goes through $\log N$ stages of distribution and $\log N$ corresponding stages of reconstruction. The figure also shows that an $N \times N$ Benes network can be constructed by placing two banyan networks back-to-back. The two banyans are called the *distribution* and the *routing* network, respectively [12], since the first distributes incoming traffic over the N links in the middle of the network – a virtual "wide" link of throughput N – and the second routes cells to the proper output link.

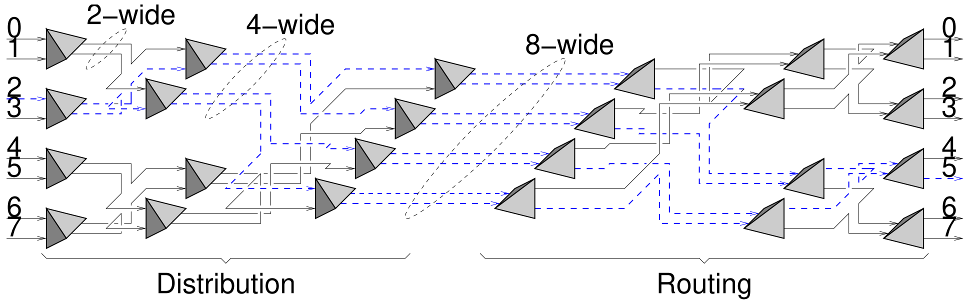

Figure 2.2: 8×8 *Benes network highlighting distribution and reconstruction of traffic* $\lambda_{2,5}$.

Non-blocking operation as above is based on (repeated) *inverse multiplexing* or *load distribution* in a balanced manner. A "poor man's" method for load distribution is to send all packets of "half" the microflows through one path, and all packets of the other half through the other path, e.g. using a pseudo-random hash function of the source-destination IP address pair to decide the path . This ensures that all packets of a given microflow follow the same route, and hence arrive in-order. The disadvantage of this method is that load distribution may not be balanced in the long run, and even worse on a short term basis, especially where the number of microflows is limited. Imbalanced load distribution will result in internal blocking in the Benes fabric, and thus we do not use this method. At the other end of the spectrum is a method for exact load distribution that resembles the *bit-sliced* processors of the 70's. Each cell is split in two units, of half the original cell (payload) size each, and each unit is sent in one of the two directions. This method is used in several commercial chip sets, but only with splitting degrees up to 8 and with carefully equalized delays through the paths [13]. This method is far from scalable, due to the fixed header and per-unit-processing overheads, and thus we do not use it.

To achieve balanced load distribution in the long run –even if not so on a very short term basis– while still operating at the cell level, a number of methods have been proposed: randomized [14], adaptive [8], 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 [11] [9]. Resequencing is an important issue in our system, dealt with in sections 2.2.1 and 3.3.

2.1.2 Internal Backpressure Protocols

Switches with multistage buffering typically use *backpressure* feedback control between these stages, *(a)* to avoid overflow of downstream buffers and *(b)* to control individual flow rates when multiple flows merge into oversubscribed resources, thus enforcing quality-of-service (QoS) guarantees.

The simplest backpressure protocol is *stop-and-go*: the upstream stage maintains a single bit of state (in total or per-flow), specifying whether downstream transmission is currently enabled or disabled. A more sophisticated protocol, which economizes on buffer memory space, is the one using *credits*: the upstream stage maintains a credit counter (in total or per-flow), specifying how many cells is is allowed to transmit in the downstream direction before new credit is received via backpressure feedback signals. The buffer space needed is $\lambda \times RTT$ (in total or per-flow), where λ is the peak rate and RTT is the round-trip time. This paper uses 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. When individual flow granularity is excessive, one can use a "compromise" solution or appropriate flow aggregation. Compromise backpressure protocols yield good performance in the usual cases, but perform badly in some worst cases; they include: wormhole virtual channels [16], a DEC proposal [17], Quantum Flow Control [18], and the ATLAS I multilane backpressure [19].

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 main tools used in this endeavor are the *merging of flows with common destination* and *hierarchical backpressure*. When multiple flows of a same priority level follow a common path to a common destination, they can be treated as a single, merged flow over the common path for purposes of buffer allocation and backpressure granularity. The reason is that cells of one flow will never need to overtake cells of another after the merge point. One (mild) disadvantage of such merging is its transient behavior when one of the flows goes from inactive to active: the "pipeline" ahead of the merge point has already been filled with cells of the other flows. Under weighted round robin (WRR) scheduling schemes, we run the danger that this pipeline empties at the rate corresponding to the weights of the old flows, while the recently activated flow may have much higher weight.

2.1.3 Hierarchical Backpressure

Hierarchical backpressure [20, section III-D], illustrated in fig. 2.3, can be used when multiple flows (preferably of the same priority level) share a common portion of their paths. This case differs from the previous case: here the flows are allowed to diverge after their common path. The flows can be treated as a single, merged flow over their common path, provided that a higher, second level of flow control feedback exists from the point of divergence (end of common path) to the point of flow merging. The common path behaves as a single, "virtual" link, and the second level flow control is over this virtual link.

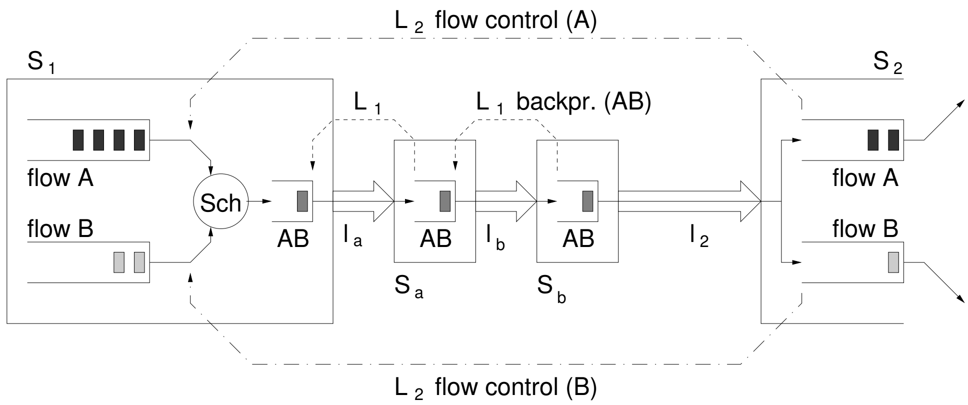

Figure 2.3: *Hierarchical flow control.*

In fig. 2.3, flows A and B have a common path, extending from switch (or stage) S_1 to S_2, through switches (or stages) S_a and S_b. Over the common path (links l_a, l_b, and l_2) the flows are treated as a merged flow "AB". At the merge point, a scheduler "Sch" decides which flow to get each next AB-cell from; it bases its decisions on *level 2 (L_2)* flow control feedback information that it receives from the divergence point, S_2. Level 1 (L_1) backpressure is at the granularity of the merged flow, AB. L_1 backpressure goes from S_b to S_a over l_b, and from S_a to S_1 over l_a. It is important to notice that no L_1 flow control is needed over link l_2: the flow into buffers A and B of switch S_2 is regulated via L_2 –not L_1– flow control.

An important application of hierarchical flow control, as drawn in fig. 2.3, is in large switches made using multistage fabrics and "line cards" that contain input buffers with virtual-output queues (VOQ). In figure 2.3, consider that S_1 is the ingress line card of (large) switch S_1, and S_2 is the ingress line card of the next downstream (large) switch S_2; S_a and S_b are the stages of the switching fabric of S_1. Links l_a and l_b are internal to the switching fabric, while link l_2 is a network (e.g. WAN) link. Under such a scenario, observe that L_1 backpressure is purely internal to the switching fabric, while L_2 is the network level flow control (hop-by-hop or end-to-end, credit- or rate-based). There is no L_1 backpressure on the network link l_2.

This is the model assumed in this paper: we deal exclusively with the flow control "L_1" *inside* the Benes fabric. We assume that this is of the credit-based backpressure type, independent of the type of flow control employed outside the fabric, in the overall network. Note from fig. 2.3 that L_1 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. The merging of multiple external flows into one of our above internal flows, performed in the scheduler "Sch" in fig. 2.3, is performed in the ingress line card of the large switch, just before entry into the Benes fabric, and that is not a topic of the present paper.

2.2 Switching Element Organization

In this section, we present flow merging schemes that reduce 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.

2.2.1 Flow Groups

As noted in sections 2.1.2 and 2.1.3, for an $N \times N$ Benes fabric, backpressure must operate at the granularity of the N^2 flows (per priority level) defined by all input-output pairs. In banyan fabrics, although the total number of flows is N^2, only N flows pass through any individual link in the fabric. In the Benes fabric, however, the traffic of every flow is distributed and sent over both "even" and "odd" subnetworks in fig. 2.1; 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). We need to reduce this number, using the flow merging techniques of sections 2.1.2 and 2.1.3.

We first consider *per-output* merging of the flows destined to the same output port of the fabric. Fig. 2.4 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. 2.4 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.)

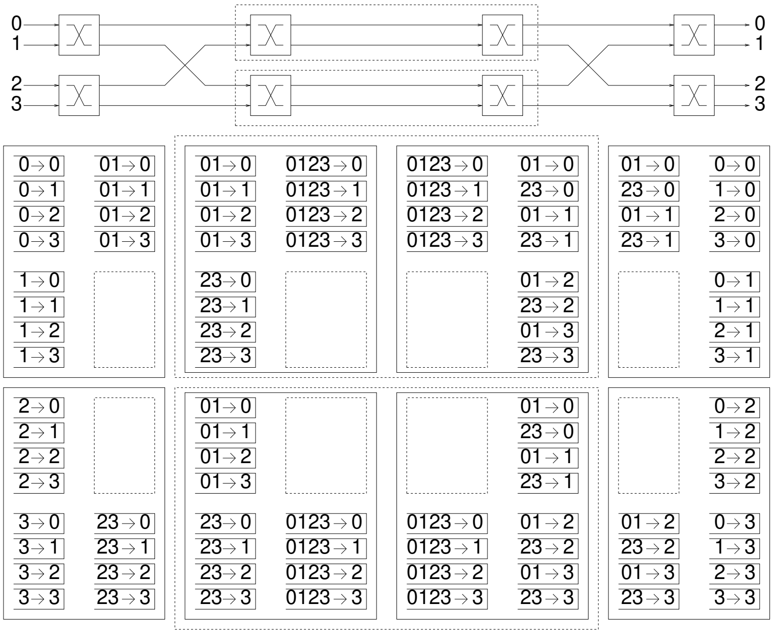

Figure 2.4: *A 4×4 fabric and the flow groups at the inputs and outputs of the switching elements for the per-output flow merging case.*

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 though 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.

An alternative method to reduce the number of flows in the center of the fabric is *hierarchical flow merging*, which follows the recursive construction of the Benes network. Specifically, the switching elements at the edges of a $K \times K$ Benes subnetwork (refer to fig. 2.1) maintain state for the K flows destined to each output or originating from each input of that subnetwork. This method reduces the number of flows even more than per-output merging, down to $N/2^k$ in stage k. However, flows destined to different *final* destinations are merged together, so *hierarchical backpressure* (section 2.1.3) is needed, which is more complicated than plain backpressure to implement. This scheme needs a total of $\log N$

levels of backpressure. Although hop-by-hop backpressure is no longer needed in the routing network, the flow control feedback delay is up to $2 \log N$ for the upper flow control level, hence buffers of size up to $2 \log N$ are needed for full link utilization, which is undesirable.

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.

2.2.2 Logical Buffer Organization

Figure 2.5 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.

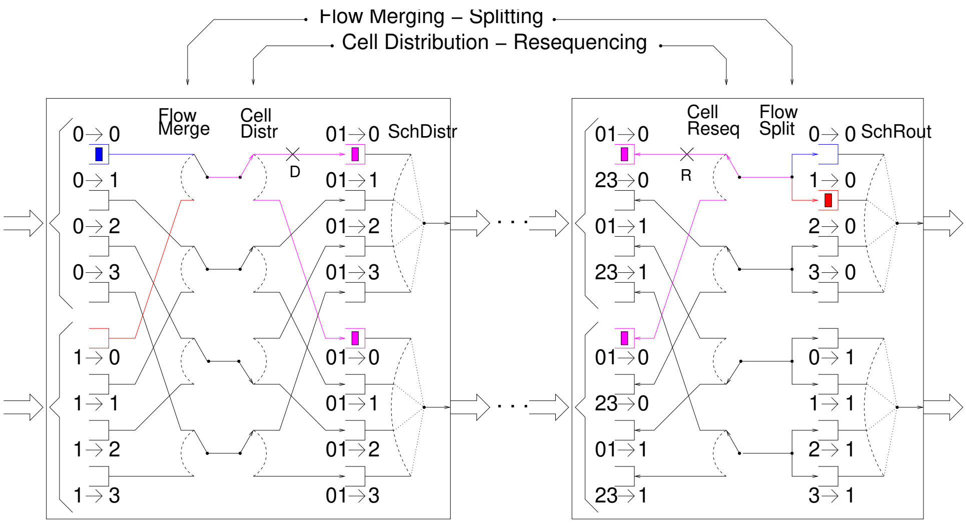

Figure 2.5: *Logical buffer organization of a distribution and the corresponding routing switching element.*

Distribution switching elements must perform flow merging and cell distribution; they can perform these tasks in either order. Routing switching elements must perform cell resequencing and flow splitting in the proper, corresponding order. Cell distribution can be performed in a number of ways; as discussed in section 3.3, it relies on per-flow state and aims to optimize per-flow criteria. Flow merging before cell distribution reduces the number of flows seen by cell distribution. The smaller the number of flows, the easier it becomes to coordinate the per-flow (local) decisions so as to optimize global criteria; also, buffer space gets reduced, as explained later in this section. Thus, we choose this arrangement, as shown in fig. 2.5.

At the inputs of the switching elements, buffers are needed per input port and per flow group, because credits for that buffer space and at that granularity must be sent to each upstream neighbor. Besides these input buffers, it is advantageous or necessary to also have output buffers, as shown in fig. 2.5. The chosen arrangement requires $2 \times P \times N$ FIFO queues per distribution switching element. If the distribution switching elements performed cell distribution before flow merging, then, each of them would need $P^2 \times N$ FIFO queues.

In the distribution switching elements (left half of the network), it is advantageous to have output buffers *(a)* in order for output schedulers to operate independently, and *(b)* for efficiency in some distribution circumstances, as explained below. Suppose that output buffers did not exist. First, assume that input buffer $0 \rightarrow 0$ contains a cell while input buffer $1 \rightarrow 0$ is empty (as in fig. 2.5), and that the cell distribution algorithm allows the cell to depart in either direction. Then, up to one but not both output schedulers of this switching element would be allowed to choose flow group $01 \rightarrow 0$ for service; hence, the two schedulers would not be able to operate in parallel. Next, assume that both input buffers $0 \rightarrow 0$ and $1 \rightarrow 0$ contain cells, and assume that the cell distribution algorithm dictates that the next-in-order cell of flow group $01 \rightarrow 0$ must depart through the top output of the switching element. Until the top-output scheduler is able to serve this next-in-order cell, it would be very hard for the bottom-output scheduler to serve flow group $01 \rightarrow 0$, although two cells exist in this flow group, because we don't quite know which cell is the second-next in order.

In the routing switching elements (right half of the network), input buffers are needed for the same reason as for the distribution switching elements, unless we know where to expect the next cell from in which case we only need one buffer slot and the credit for that buffer is sent to the upstream node from which the next cell will arrive. Each output buffer, together with its input counterpart in the downstream neighbor switch, forms a double-depth buffer pipe, which is needed for deadlock-free operation of cell resequencing under the preferred distribution methods, as will be seen in section 3.3. However, output buffers are not necessary, they could be dropped by making, at the same time, the input buffers of greater depth.

2.3 Contention Points

In this section we give an overview of the contention points in the $N \times N$ Benes fabric. We identify two reasons that cause cells to delay within the fabric: *(a)* input/output contention, and *(b)* traffic quantization.

Traffic quantization causes delays that would not be present under the fluid traffic model. The characteristics of these delays depend on the specifics of the cell distribution method. Consider an 4×4 fabric, part of which is shown in Fig. 2.6. Inefficiencies of the cell distribution method at the first stage of the distribution Banyan cause cells to delay at: *(a.1)* the `SchDistr` queues of the switching elements at stage 0 of the distribution Banyan, *(a.2)* the `SchRout` queues of the switching elements at stage 1 of the routing Banyan, and *(a.3)* the `Reseq` queues of the switching elements at stage 0 of the routing

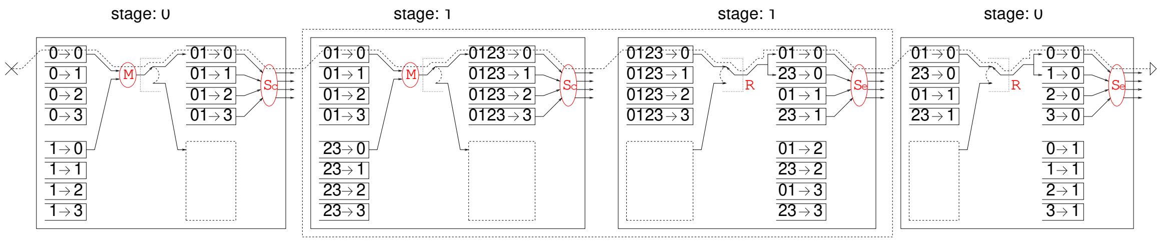

Figure 2.6: *Possible contention points for a 4×4 fabric with per-output flow merging. The figure only shows the first row of the switching elements of the fabric.* `M` *stands for* `Merge`*,* $\mathtt{S_d}$ *stands for* `SchDistr`*,* `R` *stands for* `Reseq`*, and* $\mathtt{S_r}$ *stands for* `SchRout`*.*

Banyan.

We now provide some examples that demonstrate the above cases. Suppose that two cells that belong to flows $0 \rightarrow 0$ and $1 \rightarrow 3$ arrive at the fabric at the same cell time, and the cell distribution method decides that they both be forwarded through the even Benes subnetowrk, then case *(a.1)* arises. Now, suppose that two cells that belong to flows $0 \rightarrow 0$ and $3 \rightarrow 1$ arrive at the fabric at the same cell time, and the cell distribution method decides that they are both forwarded through the even Benes subnetwork. If the two cells arrive at the output of the even subnetwork at the same cell time – which is the case if, for example, the fabric contains no other cells and the schedulers are work-conserving – then case *(a.2)* arises.

With regard to case *(a.3)*, suppose that two cells c_i and c_{i+1} of a given flow arrive at the fabric back-to-back, and the cell distribution method decides to forward c_i through the even subnetwork and c_{i+1} through the odd subnetwork. If the input traffic to the odd and even subnetworks is not identical – which will probably be the case – then cells c_i and c_{i+1} will face different delays through the two subnetworks. In order to preserve per-flow cell order, cell c_{i+1} cannot move to output 0 before cell c_i arrives from the even subnetwork. If cell c_i faces longer delay than cell c_{i+1}, then case *(a.3)* arises. In a fluid Benes network, the two Benes subnetworks would carry identical traffic, thus, they would provide identical delays to corresponding traffic units.

Queueing points caused by input/output contention are present in any switching architecture with finite buffers and/or finite switching capacity. In the discussion of contention queueing points, we assume a fluid traffic model and discard the effects of packet quantization discussed earlier in this section. First of all, note that if we follow the traffic distribution method that ensures non-blocking operation, summarized in section 2.1.1, any input traffic pattern, including those that do not satisfy output constaints, can get through the distribution Banyan to the middle stage of the fabric without delay.

With infinite buffers and no feedback, all queueing is performed at the queues of `SchRout` but the fabric is not able to enforce service ratios between flows destined to the same output. The reason being that incoming traffic reaches the middle stage without queueing delays, and by that time, a single flow per final output has been formed. With finite buffers and feedback, the output constraints cause packets to wait at the `SchRout` queues and because of the feedback, at `SchDistr` and `Merge`. The roles of `Merge`, `SchDistr` and `SchRout` in the scheduling architecture and their interactions thereof are studied in chapter 6.

Chapter 3

Cell Distribution and Resequencing

3.1 Introduction

In a Benes fabric there are N internal paths for each input-output pair, one path through each of the N middle links. The cell distribution method is responsible for choosing an appropriate path through the fabric for each incoming cell – that is, cell distribution is an instance of the dynamic, Benes routing problem. We are interested in scalable, hardware-efficient methods. Thus, we study methods of the following form: each input switching element (see Fig. 2.1) examines the incoming cells and depending on their flow (i.e. source and destination), and information maintained locally, determines which Benes subnetwork to forward the cell through.

In the rest of this chapter, we first present the class of cell distribution methods we consider, the rationale behind it, and per-flow round-robin cell distribution which belongs to this class, we then prove that the aforementioned class of cell distribution methods is actually deadlock-free.

3.2 Cell Distribution with Maximum Per-flow Imbalance of 1

The objective of the cell distribution method is to equalize the traffic load steered to the two Benes subnetworks. The method proposed in Wide Links [21] is to forward the cells of each flow *alternatingly* to the two Benes subnetworks. The above method can be implemented in a distributed fashion and is simple enough for a hardware implementation at high speed. In the absence of cell losses, cell resequencing reduces to receiving the cells of each flow alternatingly from the two subnetworks. In order to guarantee in-order cell delivery, it suffices for the routing switching elements to start delivering the cells of each flow from the same subnetwork through which the distribution switching elements started forwarding the cells of that flow. We call the above method of resequencing *coordinated round-robin*. The above method was also proposed in the context of the parallel packet switch (PPS) architecture [15] in order to eliminate the internal speedup requirement.

Consider microflow $i \rightarrow j$, the distribution switching element connected to input i and the routing switching element connected to output j. Let C^e_{ij} denote the set of cells of microflow $i \rightarrow j$ *currently* on some path of the fabric which starts, at the distribution switching element, just *after* the cell distribution point towards the even Benes subnetwork (e.g. point "D" at Fig. 2.5) and terminates, at the routing switching element, just *before* the cell resequencing point from the even Benes subnetwork (e.g. point "R" at Fig. 2.5). Let C^o_{ij} be the counterpart of C^e_{ij} for the odd Benes subnetwork. Note that the sets C^e_{ij} and C^o_{ij} are disjoint since the corresponding sets of paths are disjoint.

Theorem 1 *With per-flow round-robin cell distribution, it holds:*

$$| \, |C^e_{ij}| - |C^o_{ij}| \, | \leq 1, \ \forall \, (i, j) \tag{3.1}$$

Proof: First consider the case where both sets C^e_{ij} and C^o_{ij} are non-empty. Without loss of generality, assume that the oldest of all cells in $C^e_{ij} \cup C^o_{ij}$ is contained in C^e_{ij} and has a sequence number equal to $2 \cdot k$. The oldest cell contained in C^o_{ij} will have a sequence number equal to $2 \cdot k + 1$. Let $2 \cdot l + 1$ be the sequence number of the newest cell in C^o_{ij}. The newest cell contained in C^o_{ij} will have a sequence number equal to either $2 \cdot l$ or $2 \cdot l + 2$. Thus,

$$\begin{aligned}
|C^o_{ij}| &= \frac{(2 \cdot l + 1) - (2 \cdot k + 1)}{2} + 1 = l - k + 1 \\
|C^e_{ij}| &= \frac{(2 \cdot l) - (2 \cdot k)}{2} + 1 = l - k + 1, \ \textit{or} \\
|C^e_{ij}| &= \frac{(2 \cdot l + 2) - (2 \cdot k)}{2} + 1 = l - k + 2
\end{aligned}$$

In both cases the equation holds.

Now assume, without loss of generality, that set C^o_{ij} is empty. If C^e_{ij} is also empty, then, the equation holds. Now, let the oldest cell in C^e_{ij} has a sequence number equal to $2 \cdot k$. If C^e_{ij} contains more than one cells, then, it must also contain the cell with a sequence number equal to $2 \cdot k + 2$, because of in-order cell delivery. Morever, C^o_{ij} must contain the cell with a sequence number equal to $2 \cdot k + 1$, which is a contradiction. Thus, set C^e_{ij} contains only one cell and the equation holds. $\triangleleft$

Eq. 3.1 means that round-robin cell distribution perfectly equalizes the per-flow traffic volume among the two Benes subnetworks. However, eq. 3.1 also implies that:

$$| \sum_j |C^e_{ij}| - \sum_j |C^o_{ij}| \, | \leq N, \quad \forall \, \textit{input } i \tag{3.2}$$

$$| \sum_i |C^e_{ij}| - \sum_i |C^o_{ij}| \, | \leq N, \quad \forall \, \textit{output } j \tag{3.3}$$

Eqs. 3.2 and 3.3 mean that round-robin cell distribution does not perfectly equalize among the two subnetworks the total traffic volume coming from each input or destined to each output of the Benes network. The performance of round-robin cell distribution is analyzed in chapter 5.

Per-flow round-robin cell distribution is not the only method we study in this paper. We consider the class of cell distribution methods with maximum per-flow *imbalance* of 1: at any time, the total number of cells belonging to some flow that have been forwarded through any two paths available to that flow differs by at most 1. At the other end of the two paths, resequencing "consumes" cells in order, the resequencer essentially keeps track of the distributor with some delay; it follows that, for such distribution methods, the number of cells buffered along the two paths can differ by at most 2. We see that these distribution methods tend to equalize the load on any two paths. As proved earlier, per-flow round-robin cell distribution is such a method where the number of cells on the two paths can differ by at most 1.

3.3 Freedom from Deadlock

The Benes fabric with finite buffers, internal backpressure, per-output flow merging, cell distribution and resequencing is a distributed system with finite resources and resource sharing. In such a system, we have to make sure that deadlock situations either do not occur, or if they do occur, the system detects and resolves them. In this section, we show that for the class of cell distribution methods with maximum per-flow imbalance of 1, a deadlock situation cannot arise.

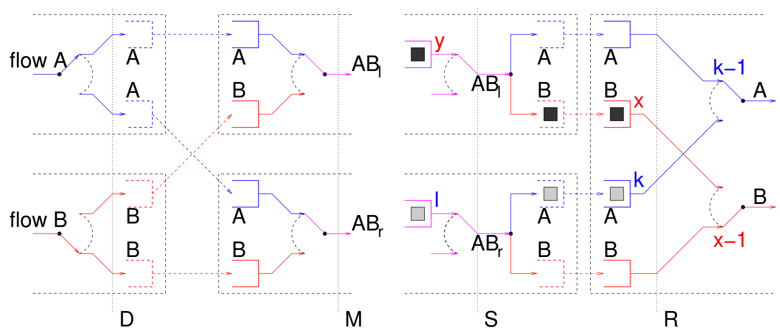

Figure 3.1: *Deadlock situation when flow merging precedes cell distribution. The participating switching elements are shown with dashed lines and the participating flows are indicated with A, B and AB. The numbers by the FIFO buffers denote the sequence number of the cell at the head of the buffer.*

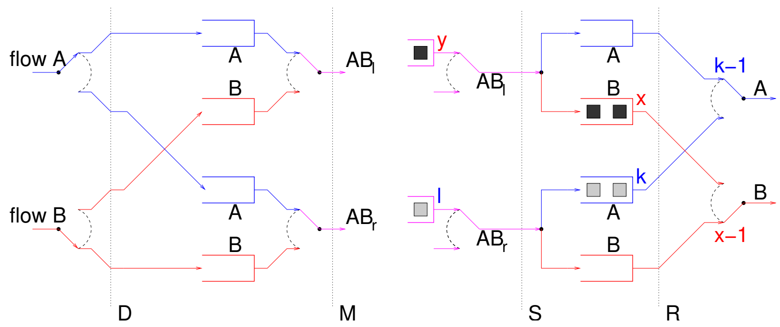

Figure 3.2: *Simplified but equivalent view of the deadlock situation shown in fig. 3.1.*

Figure 3.1 shows how a deadlock could arise in our switching elements. Figure 3.2 shows a simplified but *equivalent* view of the deadlock situation: consecutive FIFO buffer dedicated to the same flow have

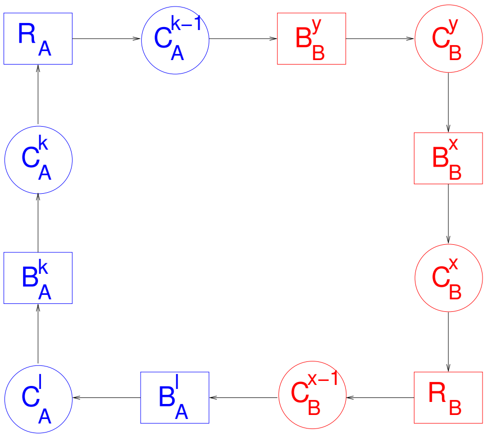

Figure 3.3: *The resource allocation graph for the deadlock situation. Circles represent cells, while rectangles represent resources which can be either buffer slots or resequencers.*

been merged into a single FIFO buffer with depth equal to the sum of the depths of the individual FIFO buffers. Let c_f^s denote the cell of flow "f" with sequence number "s", B_f^s denote the buffer slot occupied by cell c_f^s, and R_f denote the resequencer of flow "f". The deadlock situation is the following (see fig. 3.3 for the resource allocation graph): *(a)* the resequencer of flow A is waiting for cell c_A^{k-1}, *(b)* cell c_A^{k-1} is somewhere in the fabric behind cell c_B^y and it needs a buffer (B_B^y) or resequencer (R_{AB_l}) held by cell c_B^y in order to move forward – note that cell c_A^{k-1} could not be on the same path with cell c_A^k since they belong to the same flow, thus, the Benes fabric would not reorder the two cells, *(c)* cell c_B^y needs a buffer held by cell c_B^x in order to move forward, *(d)* cell c_B^x is waiting to be resequenced by R_B, and so on, so forth until the cycle closes to R_A.

The cells get through the distribution banyan facing only queueing delays. If we used resequencing at the final outputs, then the cells would get through the routing banyan facing queueing delays due to output contention, but they would face no resequencing delays within the fabric – in this case we would have to find sufficient conditions for deadlock-free operation of the resequencers at the final output. In the case of resequencing per-stage, the cells face the resequencing delays within the fabric – we now essentially find sufficient conditions for deadlock-free operation of the per-stage resequencers.

3.3.1 Basic Case

Let b_D denote the size of the distribution FIFO buffers, and b_R denote the size of the routing FIFO buffers shown in fig. 3.2.

Theorem 2 *If no cells are lost, $b_D = 1$, $b_R = 2$ and cell forwarding is subject to hop-by-hop credit-based flow control, then any cell distribution method with maximum per-flow imbalance of 1 is deadlock-free.*

Proof: Let L_f^s denote the time slot at which cell c_f^s crosses line "L" in fig. 3.2. The cell distribution method and the fabric operation impose limitations on the set of cells that can be active on the paths available for flows A and B, and the ordering between various values of L^s_f for lines D,M,S,R. With regard to the set of active cells, there are four cases for flow A shown in fig. 3.4, the cases for flow B are analogous. The ordering relations are of four types and are listed below:

- *Direction:* $(D^i_f < M^i_f < S^i_f)$
- *Distribution:* $(D^i_f < D^{i+1}_f)$
- *Backpressure:* $(M^i_f < D^j_f)$ *for every cells* c^i_f *and* c^j_f *with* $i < j$ *which were both forwarded through the same path*
- *Flow-Order:* $(M^i_f < M^j_f \iff S^i_f < S^j_f)$ *for every cells* c^i_f *and* c^j_f *which were both forwarded through the same path*

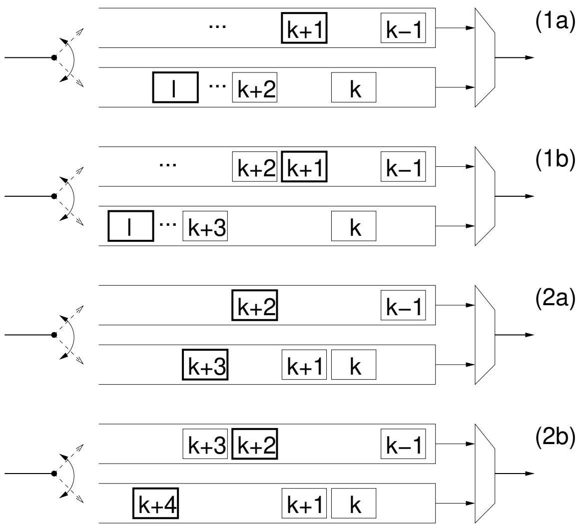

Figure 3.4: *Active cells allowed by cell distribution methods with maximum per-flow imbalance of 1.*

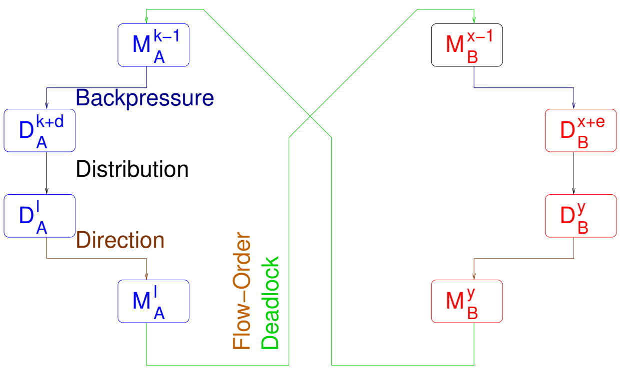

Figure 3.5: *The time-order graph. The nodes of the graph represent events of the form* L^s_f *and the arcs represent ordering relations between the events. The direction of an arc is from the older to the newer event.*

Using the above relations, we can partially construct the time-order graph shown in fig. 3.5, specifically relations ($M_A^{k-1} < M_A^l$) and ($M_B^{x-1} < M_B^y$). Cell c_A^{k+d} refers to the *first* cell of flow A after cell c_A^{k-1} on the same path with cell c_A^{k-1}. Sequence number $k+d$ corresponds to either $k+1$, cases *(1a)* and *(1b)*, or $k+2$, cases *(2a)* and *(2b)*. Cell c_A^l refers to the *second* cell of flow A after cell c_A^k on the same path with cell c_A^k. Sequence number l also differs in each of the four cases of fig. 3.4. The important property is that l is greater than $k+d$, and this property can only be guaranteed in all of the four cases only if the size of the routing FIFO buffers is 2 slots. Similarly for flow B.

If we assume that a deadlock arises, then the following ordering relations also hold: ($M_B^y < M_A^{k-1}$) and ($M_A^l < M_B^{x-1}$). The deadlock ordering relations cause a cycle in the time-order graph, which is a contradiction. Thus, a deadlock situation cannot arise.

$\triangleleft$

Note that the proof does not assume any special properties for the scheduling discipline at the flow merging point. However, it assumes that no cells are lost due to electrical noise within the fabric. This is definitely an unrealistic assumption and a real system would have to employ robust resequencing protocols as the ones described in [9]. The proof of the basic case extends easily for the case of more than two participating flows. With regard to per-flow round-robin cell distribution, note that it remains deadlock free even with routing buffers of size 1, since it only allows case *(1a)* of fig. 3.4 to arise.

3.3.2 Extensions

The proof continues to hold in the general cases of *(a)* distribution FIFO buffers of size b_D slots each, where $b_D > 1$ and *(b)* switching elements of size $P \times P$, where $P > 2$. The important observation is that the ordering relations $M_A^{k-1} < M_A^l$, and $M_B^{x-1} < M_B^y$, shown in fig. 3.5, can be guaranteed *independently* for each flow, and depend only on the properties of the cell distribution method and the available distribution buffer size b_D.

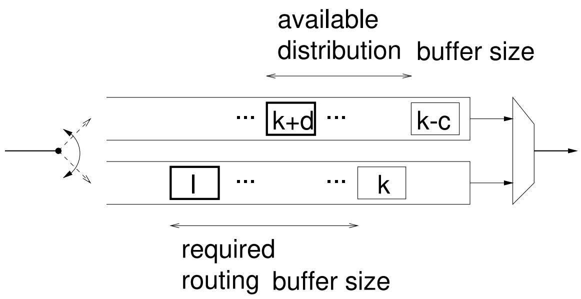

Figure 3.6: *Required size of routing FIFO buffers in order to ensure deadlock free operation.*

Consider fig. 3.6 and assume that cell c_A^{k-c} is the next cell to be resequenced, cell c_A^{k+d} is the *first* cell of flow A after cell c_A^{k-c} on the same path with cell c_A^{k-c}, and cell c_A^l is the cell of flow A with the *smallest* sequence number *greater* than $k+d$ on the same path with cell c_A^k. The relation $M_A^{k-c} < M_A^l$

holds in this case provided that we choose b_R large enough so that sequence number l is *greater* than sequence number $k+d$ in all cases allowed by the cell distribution method for the given buffer size b_D.

To sum up, for a given cell distribution method and distribution buffer size b_D, we can choose b_R so that the Benes fabric is deadlock free. In the special case of maximum per-flow imbalance of 1 and $b_D = 2$, the required b_R is 3. With regard to per-flow round-robin cell distribution, the required b_R is equal to b_D for any value of b_D.

Chapter 4

Simulation Results

4.1 Introduction

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 each buffer shown in fig. 2.5 has a size of 1 cell, except for the input buffers of the routing switching elements which have a size of 2 for a specific cell distribution method.

We simulated the switch under smooth, bursty, and hotspot traffic. Smooth traffic consisted of Bernoulli arrivals with uniformly distributed destinations. For bursty traffic, each source alternatingly produces a burst of cells (all with the same destination) followed by an idle period of empty cells; the bursts and idle periods contain a *geometrically distributed* number of cells. 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). The bursty traffic model is described in detail in appendix B.1. 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); for example, most of the reported results are for a 64×64 fabric made of 4×4 switching elements, hence the fabric has $2 \cdot \log_4 64 = 2 \cdot 3 = 6$ stages, and the number subtracted is 6 cell times. In our simulation model, it takes 1 cell time for a cell to traverse 1 stage of the fabric, in an otherwise idle system, with the cell moving at top speed. Thus, by subtracting the fabric length from the actual delay, we report the sum of all queueing delays for the cells. 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 ran each simulation 10 times and then computed the sample mean and the corresponding confidence interval for the measured delay. With regard to average delay, the 95% confidence intervals were well below 5% in all cases but one case for load equal to 99% where the confidence interval was 7.1%.

As a means to get an indication regarding the 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*, in the sense of section 2.1.1, 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.

4.2 Cell Distribution Methods and Comparison with OQ and iSLIP

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, 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. We also performed simulations with larger buffer sizes, up to 4, which allow more "slack" in the two paths, and found that performance is *insensitive* to this parameter. The results are shown in fig. 4.1, for uniformly destined traffic, and in fig. 4.2, for traffic in the presence of hot spots.

Under smooth (Bernoulli) traffic, the cell distribution method does make some difference: imbalance count (PerFlowIC) yields 30% to 60% lower delay when compared to round-robin distribution (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.

By comparing the delays in fig. 4.2 to those in fig. 4.1, 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%.

Figures 4.2 and 4.1 also show, for comparison, the delay of the ideal output-queued (OQ) switch

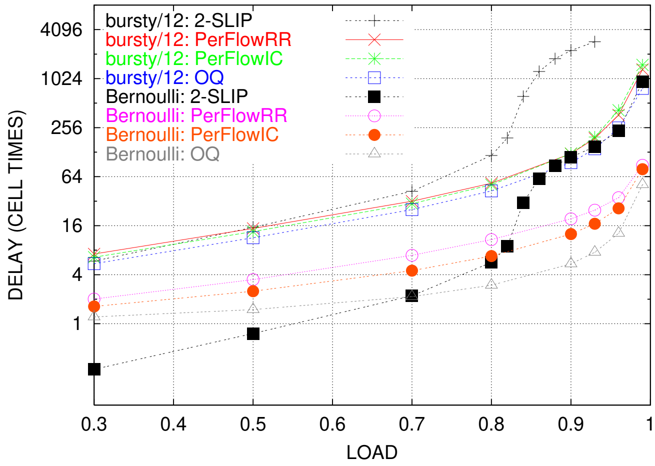

Figure 4.1: *Delay versus load for uniform destinations; 64×64 fabric made of 4×4 elements; upper curves: bursty/12 traffic; lower curves: Bernoulli traffic; ideal output queueing (OQ) also shown for comparison.*

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 25% to 50% 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.

Lastly, we compare the performance of the Benes fabric with that of a crossbar with VOQ's and the 2-SLIP crossbar scheduling algorithm [22] [1]. We see that, for loads under 70%, the delay for 2-SLIP is small, compareable to that of the delay through the Benes fabric; in the case of smooth traffic it even gets less than 1, this is because the crossbar is unbuffered, thus, the cells need not wait for 1 cell time within the crossbar. As the load gets higher, around 80%, the delay for 2-SLIP increases considerably, and for bursty traffic it gets 14 to 18 times worse than the delay through the Benes fabric.

4.3 Fabric Size Dependence of Performance

One of the advantages of the proposed architecture is than 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. 4.3, and they show that the 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

[1] For the performance simulations for the 2-SLIP algorithm, we used the SIM simulator from Stanford University. The traffic model for bursty traffic 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%.

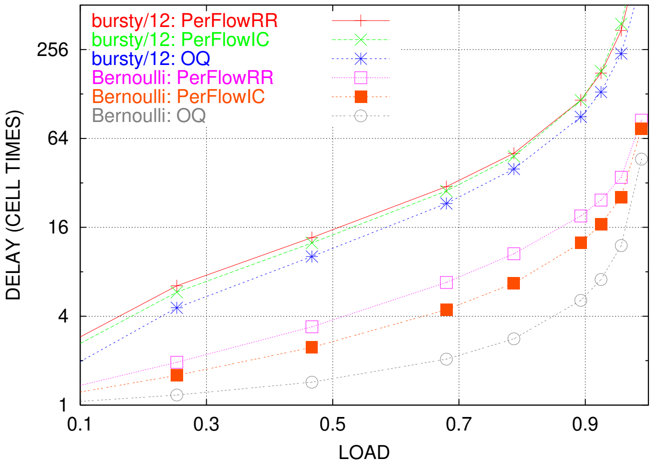

Figure 4.2: *Delay of non-hotspot destinations in the presence of hotspot/4 traffic; horizontal axis is the load to non-hotspot outputs; other parameters as in fig. 4.1.*

this case, for loads under 70%, the delay remains unaffected with increasing fabric size, but for higher loads it gets approximately 4 times worse with increasing fabric size.

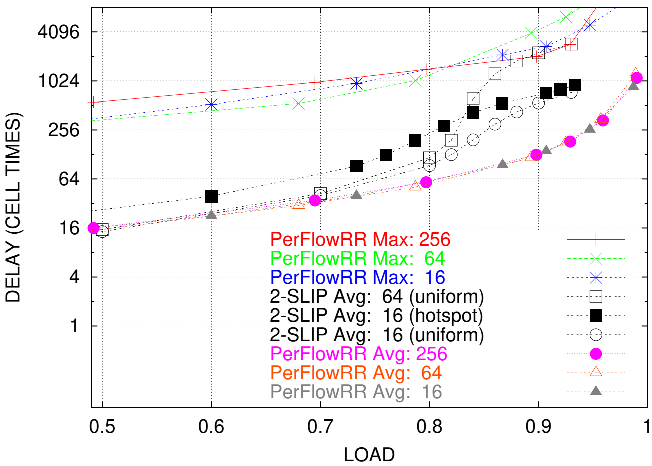

Figure 4.3: *Performance for various fabric sizes,* 16×16 *to* 256×256*: average delay and maximum delay versus load, under bursty traffic in the presence of hot spots. The results are for the PerFlowRR cell distribution method.*

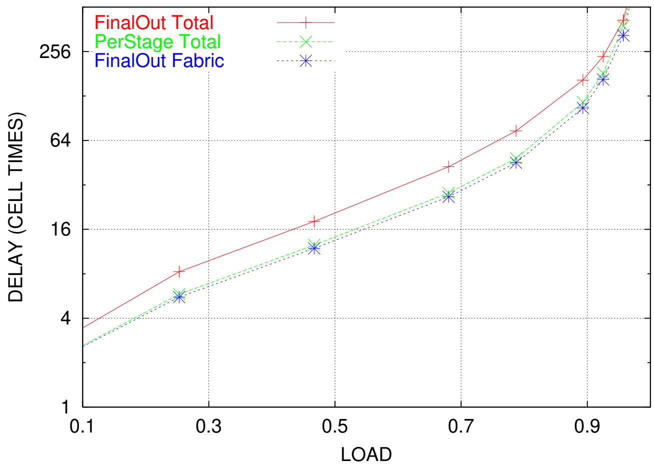

Figure 4.4: *Average delay under different resequencing methods; bursty traffic in the presence of hot spots. The results are for the PerFlowIC cell distribution method.*

4.4 Alternative Cell Resequencing Methods

As discussed in section 2.2.1, 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 remained regarding performance: 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 4.4 shows the average delay under the two resequencing methods; input traffic is bursty/12 and hotspot/4, as in section 4.3. 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.

Chapter 5

Performance Analysis

5.1 Introduction

The result of the imbalance described by eqs. 3.2 and 3.3 is internal delays which would not be present under a fluid traffic model or a more sophisticated cell distribution method. Specifically, inefficiencies of the cell distribution method at some stage of the distribution banyan result in delays for access at *(a)* the output links of the distribution switching elements at that stage, called *input* delay in this report, *(b)* the input links of the routing switching elements at the corresponding routing stage, called *output* delay in this report. However, as noted in [23], output delay is impossible to eliminate when input switching elements operate independently, as it is the case with a distributed algorithm. The arguement for this, along with the symmetrical arguement for input delay, are summarized below:

- `output` delay: two input switches forward, at the same cell time, two cells through the same subnetwork and the two cells are destined to the same output switch.
- `input` delay: two output switches choose to serve, at the same cell time, two cells that originate from the same input switch and were forwarded through the same subnetwork.

In this section we analytically show that these internal delays do not result in throughput limitations. However, for this result, we consider a simplified model of the Benes fabric without internal backpressure. Specifically, in this analysis, we consider a model of the Benes fabric with *(a)* per-output flow merging, *(b)* per-flow round-robin cell distribution, *(c)* infinite buffers in the middle stage of the fabric, and *(d)* no backpressure.

First of all, note that in a fluid model with infinite buffers, no feedback, and a traffic distribution method that equally divides the per-flow traffic among the outputs of each distribution switching element, any input traffic pattern passes through the distribution banyan to the middle stage of the fabric without queueing delays. In a packet system, though, incoming cells face finite queueing delays through the distribution banyan because of input delay. Also, note that such a fluid model when combined with per-output flow merging is unable to enforce service ratios between flows destined to the same output. The

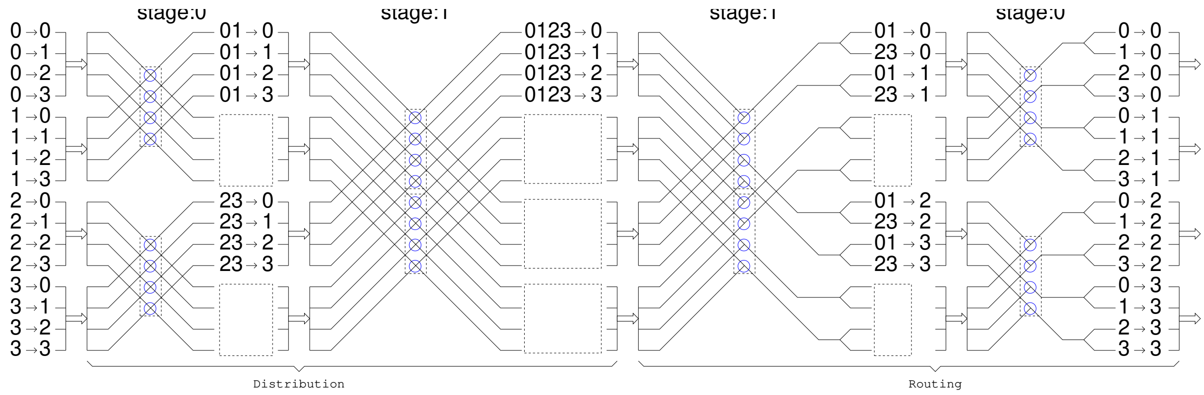

Figure 5.1: *A 4×4 Benes fabric and all the flows through the fabric. The blue circles within the distribution banyan denote flow merging and cell distribution functionality, while the blue circles within the routing banyan denote cell resequencing and flow splitting functionality. Blue circles grouped together constitute a single switching element.*

reason being that incoming traffic reaches the middle stage without queueing delays, and by that time, a single flow per final output has been formed. However, it is possible to isolate flow groups destined to different outputs from each other: it suffices for the schedulers that feed the links at the heart of the fabric to *statically* and *equally* divide the capacity of the links they feed among the N flow groups they serve. Then, the traffic per final output that enters the routing banyan is no more that 1 cell per cell time over any time scale, and traffic passes through the routing banyan without contention.

For the above reason, in this analysis which takes into account the effects of cell distribution, we characterize the performance of the $N \times N$ Benes fabric with regard to the performance of an $N \times N$ output-buffered FIFO switch. The approach we use and the techniques we apply in the analysis are influenced by the ones presented in [24] which is an application of the network calculus theory.

$A(t)$	Cummulative number of cell *arrivals* in $[0, t)$
$B(t)$	Cummulative number of cell *departures* in $[0, t)$
$C(t)$	Cummulative number of *service chances* assigned to the queue in $[0, t)$
$q(t)$	Backlog of the queue at time t

Table 5.1: *Notation for the variables that describe the state of a FIFO queue.*

The basic equation of network calculus is now explained. Consider a FIFO queue served by a work-conserving scheduler with time-varying capacity. The state of the above system can be described by a set of variables shown in Table 5.1. For such a FIFO queue it holds:

$$
\begin{aligned}
q(t) &= \max_{0 \le s \le t} [A(t) - A(s) - (C(t) - C(s))] \\
B(t) &= \min_{0 \le s \le t} [A(s) + C(t) - C(s)]
\end{aligned}
$$

When the scheduler serves the FIFO queue with constant rate r, it is $C(t) = r \cdot t$, thus:

$$\begin{aligned} q(t) &= \max_{0 \le s \le t} [A(t) - A(s) - r \cdot (t-s)] \\ B(t) &= \min_{0 \le s \le t} [A(s) + r \cdot (t-s)] \end{aligned}$$

The equations for the constant-rate case are expressed graphically in Fig. 5.2.

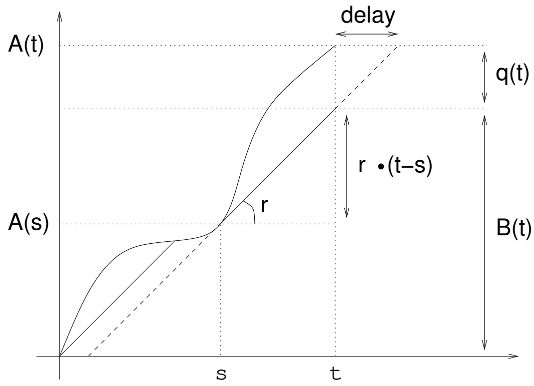

Figure 5.2: *Graphical expression of the backlog $q(t)$ of a FIFO queue served at a constant rate r.*

5.2 Distribution Banyan

We first consider input delays. The result of this section is similar to the result of [15, Lemma 3]. However, we repeat it here in the context of the Benes fabric in order for the report to be self-contained. We use the notation shown in Table 5.2.

$A_{i,j}$	Cell arrivals at fabric input i destined to fabric output j
A_j	$\sum_{i=0}^{N-1} A_{i,j}$
$A^{D(k)}_{i_k,p_k,j}$	Consider the switching element with index i_k at stage $k-1$ of the distribution banyan. This variable is the number of cell arrivals at the queue for final output j at output port p_k of that switching element.
$A^{D(k)}_{i_k,p_k}$	$\sum_{j=0}^{N-1} A^{D(k)}_{i_k,p_k,j}$
$d^{D(k)}$	Maximum delay of a cell through the output buffer of a switching element at stage $k-1$ of the distribution banyan

Table 5.2: *Notation for the variables that describe arrivals at the output queues of the distribution switching elements. The notation for the rest of the variables (B, C, b) is analogous.*

We employ jitter control or delay equalization similar to [24]. That is, cells are kept within the input queues of the distribution switching elements before they are forwarded to the output queues, so that all cells face the same delay $d^{D(k)}$ from the time they enter the output queue of stage $k-1$ until they enter the output queue of the next stage of the distribution banyan. Jitter control effectively prevents cells that arrive at the fabric in a given cell time and are destined to a given fabric output to overtake cells that are destined to the same output but arrived in previous cell times. That is, jitter control prevents reordering between cells that are destined to the same output.

For the system described above it holds:

$$A^{D(k+1)}_{i_{k+1},p_{k+1},j}(t) \leq \lceil \frac{\sum_{i_k} B^{D(k)}_{i_k,p_k,j}(t - d^{D(k)})}{P} \rceil \tag{5.1}$$

$$A^{D(1)}_{i_1,p_1,j}(t) \leq \lceil \frac{\sum_i A_{i,j}(t)}{P} \rceil \tag{5.2}$$

$$A^{D(k+1)}_{i_{k+1},p_{k+1},j}(t) \leq \lceil \frac{\sum_{i_k} A^{D(k)}_{i_k,p_k,j}(t - d^{D(k)})}{P} \rceil \tag{5.3}$$

Some comments about eq. 5.1 follow: *(a)* i_k $(0 \leq i_k < \frac{N}{P})$ is the index of a switching element within stage $k-1$ of the distribution banyan, while p_k $(0 \leq p_k < P)$ is the index of an output port of some distribution switching element. The pair i_k, p_k effectively identifies an output link of a switching element at stage $k-1$ of the distribution banyan. *(b)* i_k depends on i_{k+1} and runs over the P switching elements at stage $k-1$ of the distribution banyan whose p_k^{th} output link is an input link to the switching element with index i_{k+1} at stage k of the distribution banyan. *(c)* The time shift $(t - d^{D(k)})$ is due to jitter control. The equation assumes that $d^{D(k)}$ is finite: we will soon prove an upper bound on $d^{D(k)}$ for all values of k. *(d)* The sum over i_k corresponds to flow merging, while the division by P corresponds to round-robin cell distribution: we first merge and then distribute, thus, we first sum and then divide. With regard to eq. 5.2, distribution switching elements at the first stage of the fabric do not perform jitter control and i runs over P input links of the fabric. Eq. 5.3 is implied by eq. 5.1 and $B^{D(k)}_{i_k,p_k,j}(t) \leq A^{D(k)}_{i_k,p_k,j}(t)$.

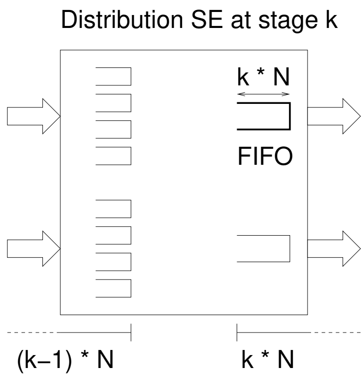

Figure 5.3: *Simplified model of the distribution switching element at stage k.*

Theorem 3 *Assume that the schedulers at the outputs of the distribution switching elements (i.e.* `SchDistr`*) perform FIFO scheduling. Then, for* $(1 \leq k \leq L = \log_P N)$ *it holds:*

$$q^{D(k)}_{i_k,p_k}(t) < k \cdot N \tag{5.4}$$

$$B^{D(k)}_{i_k,p_k}(t + k \cdot N) \geq A^{D(k)}_{i_k,p_k}(t) \tag{5.5}$$

$$d^{D(k)} \quad < \quad k \cdot N \tag{5.6}$$

Proof: See appendix A.1. ◁

Thm. 3 provides an upper bound for the worst-case delay of a cell through the distribution banyan. and the maximum required buffer space at the outputs of the distribution switching elements, Moreover, this bound hopefully increases linealrly, not exponentially, with stage number k.

$q^{D(k)}_{i_k,p_k}(t) \quad \leq \quad (1-\frac{1}{P}) \cdot N$	
$q^{D(k)}_{i_k,p_k}(t) \quad \leq \quad (1-\frac{1}{P}) \cdot P \cdot N$	cell distribution before flow merging
$q^{D(k)}_{i_k,p_k}(t) \quad \leq \quad (1-\frac{1}{P}) \cdot \frac{K}{P}$	hierarchical flow merging

Table 5.3: *Bounds for backlog in distribution switching elements without jitter control.*

With regard to backlog, we can also prove the relations shown in Table 5.3 for the case without jitter control. In the last relation, K represents the size of the Benes subnetwork for which the distribution switching element we are considering is an input switch. The above equation means that, for hierarchical merging, the input delay at the middle stages of the fabric is significantly smaller than the delay for per-output merging.

Let $L = \log_P N$, then the upper bound for the worst-case delay through the distribution banyan is in the order of $N \cdot L^2$ for the case with jitter control. The upper bound without jitter control is in the order of $N \cdot L$, but, as explained in [25], dropping jitter control may cause the Benes fabric to delay a cell in the order of N^2 more than the shadow FIFO switch.

5.3 Middle Stage

Middle stage denotes the last stage of the distribution banyan which feeds the N links at the heart of the fabric, see Fig. 5.1. All the distribution switching elements participating in the middle stage maintain the same set of output queues: one queue for each of the N final outputs of the fabric.

Let:

$$q_j(t) = \max_{0 \leq s \leq t} [A_j(t) - A_j(s) - 1 \cdot (t-s)]$$

Then, $q_j(t)$ is the backlog at time t at output j of the $N \times N$ output-buffered FIFO switch whose input traffic is identical to that of the Benes fabric.

Theorem 4 *Assume that the schedulers which feed the links at the heart of the fabric* statically *and* equally *divide their capacity among the N queues they serve. Also, let $L = \log_P N$, $d^D = \sum_{k=1}^{L-1} d^{D(k)}$ and $d^M = (L+2) \cdot N$. Then it holds:*

$$q^{D(L)}_{i_L,p_L,j}(t) \quad < \quad \frac{q_j(t-d^D)}{N} + L + 1 \tag{5.7}$$

$$B^{D(L)}_{i_L,p_L,j}(t + q_j(t-d^D) + d^M) \quad \geq \quad A^{D(L)}_{i_L,p_L,j}(t) \tag{5.8}$$

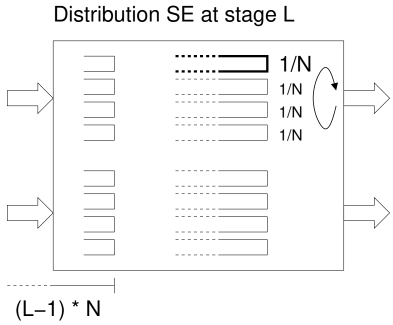

Figure 5.4: *Simplified model of the distribution switching element at the middle stage.*

Proof: See appendix A.2. ◁

Combining eq. 5.5 and eq. 5.8, we get Table 5.4. Table 5.4 denotes that some cell that arrives

Fabric input	t_0
Middle stage buffer	$t_0 + d^D$
Middle link	$(t_0 + d^D) + q_j((t_0 + d^D) - d^D) + d^M$ $= t_0 + q_j(t_0) + d^D + d^M$

Table 5.4: *Delay faced by a cell from the fabric input to the middle link.*

at the fabric input at time t_0 will arrive at some middle stage buffer *exactly* at time $t_0 + d^D$ – this is an exact time because of jitter control –, and will pass through some middle link not later that time $t_0 + q_j(t_0) + d^D + d^M$. Note that the delay of the same cell from the input to the output of the shadow FIFO switch is between $t_0 + q_j(t_0) - (N - 1)$ and $t_0 + q_j(t_0)$ by definition of the FIFO scheduling discipline. Thus, the delay of every cell from the fabric input to some middle link is larger than the delay of the same cell through the shadow $N \times N$ output-buffered FIFO switch by at most a term of $d^D + d^M + N = O(N \cdot L^2)$. Let us note that the corresponding delay term in [24] is in the order of $O(N^2)$. The reason for the reduced delay overhead is that the Benes fabric allows for more flexible schedules. These schedules are enabled by the buffering in the swithcing elements and can still be computed in a distributed fashion.

5.4 Routing Banyan

In this section, we analyze the delay faced by cells through the $\log_P N$ stages of the routing banyan. We use the notation shown in Table 5.5. With regard to $A^{R(k),rsq}_{i_k,q_k,j}$, note that, although the routing switching elements maintain per-flow queues for resequencing, we collectively count the cells of all flows destined to the same final output.

Let $* \rightarrow j$ denote a flow set that comprises of all the flows destined to final output j and pass through a given switching element. In the analysis, we assume that the schedulers at the outputs of the routing

$A^{R(k),rsq}_{i_k,q_k,j}$	Consider the switching element with index i_k at stage $k-1$ of the routing banyan. This variable is the number of cell arrivals at the resequencing queue for final output j at input port q_k of that switching element.
$A^{R(k)}_{i_k,p_k,j}$	This variable is the number of cell arrivals for final output j at output port p_k of that switching element.
$d^{R(k)}$	Maximum delay of a cell through the output buffer of a switching element at stage $k-1$ of the routing banyan.

Table 5.5: *Notation for the variables that describe arrivals at the queues of the routing switching elements. The notation for the rest of the variables (B, C, b) is analogous.*

switching elements *statically* and *equally* divide their capacity among flow sets $* \to j$. Note that the above schedulers could also perform FIFO scheduling, the reason being that the output constraints have already been enforced by the schedulers at the middle stage of the fabric. We chose static scheduling to simplify the analysis. We also make a simplifying assumption similar to jitter control that requires global knowledge, this assumption is explained within the proof of the theorem. Note that the inputs of the routing switching elements need to consider individual flows rather than flow sets in order to perform resequencing.

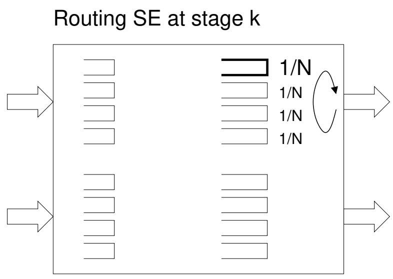

Figure 5.5: *Simplified model of the distribution switching element at stage k.*

Theorem 5 *Let $d^{mid} = d^D + d^M + N$. Assume that the schedulers at the outputs of the routing switching elements (i.e.* `SchRout`*)* statically *and* equally *divide their capacity among flow sets $* \to j$. Then, for $(L = \log_P N \geq k \geq 1)$ it holds:*

$$\bigcup_{i_L, q_L} B^{R(L),rsq}_{i_L,q_L,j}(t' + d^{mid}) \supseteq B_j(t') \tag{5.9}$$

$$d^{R(k)} = 2 \cdot k \cdot N + P^k \tag{5.10}$$

$$\bigcup_{i_k, p_k} B^{R(k)}_{i_k,p_k,j}(t' + d^{mid} + \sum_{s=L}^{k} d^{R(s)}) \supseteq B_j(t') \tag{5.11}$$

In the above equation, i_L, q_L run over all input ports at stage $L-1$ of the routing banyan, while i_k, p_k run over all output ports at stage $k-1$ of the routing banyan.

Proof: See appendix A.3. $\triangleleft$

Thm. 5 denotes that a cell passes through the resequencing buffers at stage $L-1$ of the routing banyan with an additional delay of at most d^{mid} with regard to the delay through the shadow FIFO switch, and passes through the routing banyan with an additional delay of at most d^R, where

$$d^R = \sum_{s=L}^{1} d^{R(s)} = \sum_{s=L}^{1}(2 \cdot s \cdot N + P^s) = 2 \cdot N \cdot \sum_{s=L}^{1} s + \sum_{s=L}^{1} P^s = O(N \cdot \log_P^2 N) + O(N)$$

Thus, the additional delay of every cell through the Benes fabric with regard to the delay through the shadow FIFO switch is:

$$\begin{aligned}
d^D + d^M + d^R + N &< (N \cdot \sum_{s=1}^{L-1} s) + (L+2) \cdot N + (2 \cdot N \cdot \sum_{s=L}^{1} s + \sum_{s=L}^{1} P^s) + N \\
&= (N \cdot \sum_{s=1}^{L} s - L \cdot N) + L \cdot N + 2 \cdot N + (2 \cdot N \cdot \sum_{s=1}^{L} s + \sum_{s=1}^{L} P^s) + N \\
&= 3 \cdot N \cdot \sum_{s=1}^{L} s + \sum_{s=1}^{L} P^s + 3 \cdot N \\
&= 3 \cdot N \cdot \frac{L \cdot (L+1)}{2} + P \cdot \frac{N-1}{P-1} + 3 \cdot N \\
&< \frac{3}{2} \cdot N \cdot L^2 + \frac{3}{2} \cdot N \cdot L + 2 \cdot N + 3 \cdot N \\
&= \frac{3}{2} \cdot N \cdot L^2 + \frac{3}{2} \cdot N \cdot L + 5 \cdot N
\end{aligned}$$

Chapter 6

Input/Output Contention Resolution Points

In this section we identify the points of the fabric where contention is resolved, i.e. the points where flows competing for the capacity of the fabric meet with each other. We consider an artificial packet system that discards the effects of cell distribution, which we call the "bit-slicing" model.

6.1 The Bit-Slicing Model

In the bit-slicing model, traffic distribution is performed according to the following method: each distribution switching element equally divides the bits of every incoming cell among its two outputs. As noted in sec. 3, this model assumes the existence of links and switching elements that can operate up to $N\times$ faster than the input/output links of the fabric. Under this model, the flow groups *and* the traffic through the P outputs of a given distribution switching element are identical. Thus, if we collapse the P outputs of every distribution switching element into a signle output, then the distribution switching elements can be considered as having P input links and one output link whose throughput is $P\times$ the throughput of each input link. Correspondingly, the routing switching elements can be considered as having one input link and P output links each one having $\frac{1}{P}\times$ the throughput of the input link. The bit-slicing model allows us to focus on the contention resolution points of the fabric, while avoiding the complications of cell distribution at this point of the analysis.

The bit-slicing model is still a packet system with finite buffers, hop-by-hop credit-based flow control, and per-output flow merging. Fig. 6.1 shows a subset of the flow groups traversing an 8×8 fabric and the corresponding switching elements. The flow mergers and splitters are denoted with blue circles within the distribution and routing switching elements, respectively. Note that their is no need for cell resequencing in this case, thus, there is no need for flow splitting. However, we use flow splitting in order for the bit-slicing model and the actual Benes fabric to have identical flow groups at corresponding

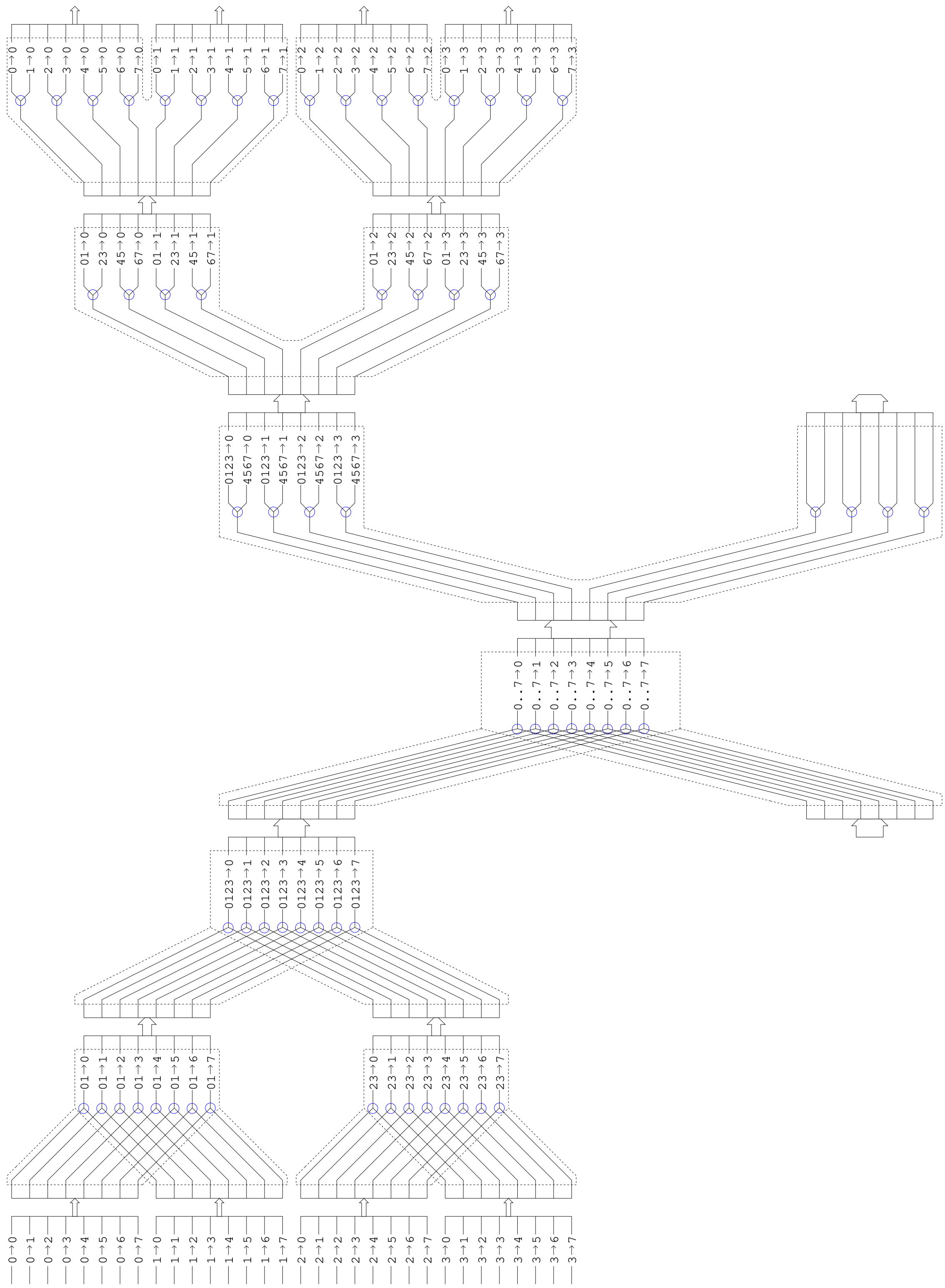

Figure 6.1: *The bit-slicing model. The figure shows a subset of the flows through an 8×8 fabric with per-output flow merging. The distribution and routing switching elements are identified with dashed lines.*

switching elements. We also assume that the distribution switching elements only maintain logical cell queues at their inputs, and the routing switching elements only at their outputs.

We now present the role of the active components of the switching elements in allocating the capacity of the fabric, and thus delay and bandwidth guarantees, to the $N \times N$ competing flows. In general, allocating the capacity of a switching system reduces to enforcing input and output constraints, and resolving input and output contention. Among the single-stage, single-module switching fabrics, the buffered crossbar architecture [26] is the closest one to the Benes fabric. The buffered crossbar architecture consists of virtual output queues (VOQ) at the inputs followed by a single stage of $N \times N$ buffers, one buffer per input-output pair. Moreover, there are $N + N$ schedulers, one at each of the inputs and outputs of the fabric. Input constraint enforcement and input contention resolution are performed by the input schedulers, while output contention resolution and output constraint enforcement are performed by the output schedulers.

6.2 Output Constraints and Output Contention

We start with output constraint enforcement and output contention resolution. Suppose that flows $0 \to 0$ and $1 \to 0$ are the only active flows in the fabric and both are continuously backlogged. The two flows first meet at the merging point (at the first stage) of the distribution Banyan. If all schedulers in the fabric are work-conserving, then the output constraint is enforced at the output (of the last stage) of the routing Banyan. Because of finite buffers and backpressure, the output constraint is finally enforced at the merging point, whose output rate is reduced to match the capacity of the output link. After rate matching, the `Merge` is responsible for allocating the capacity of the output link to the competing flows.

In general, there is one tree of `Merge`'s, within the distribution Banyan, for each output link of the fabric. The `Merge` tree allocates the capacity of the output link to the N competing flows in a hierarchical fashion as the competing flows meet at the nodes of the tree. Note that in the actual Benes fabric the `Merge` trees correspond to `Merge` butterflies.

If all schedulers in the fabric are work-conserving, then the output constraints are enforced at the outputs of the routing switching elements and are propagated, through backpressure, to the heads of the `Merge` trees. However, the above is not the only option for output constraint enforcement. We could also enforce output constraints directly at the outputs of the last stage of the distribution Banyan, see Fig. 6.2 *(i)*, or within each stage of the distribution Banyan, see Fig. 6.2 *(ii)*. Note that both options still allow any subset of the flow set $\{\ 0 \to 0,\ \ldots,\ 7 \to 0\ \}$ to fully occupy the capacity of output 0 – i.e. they only block traffic in excess of an output link's capacity.

In the option shown in Fig. 6.2 *(i)*, the feedback to the nodes of the `Merge` butterflies is faster, and the routing Banyan is free-flowing: there is no backpressure between the stages of the Banyan. In addition to the above, the option shown in Fig. 6.2 *(ii)* reduces the number of operations per FIFO queue performed in each cell time. However, both options proactively insert the idle cell times which may turn out to be

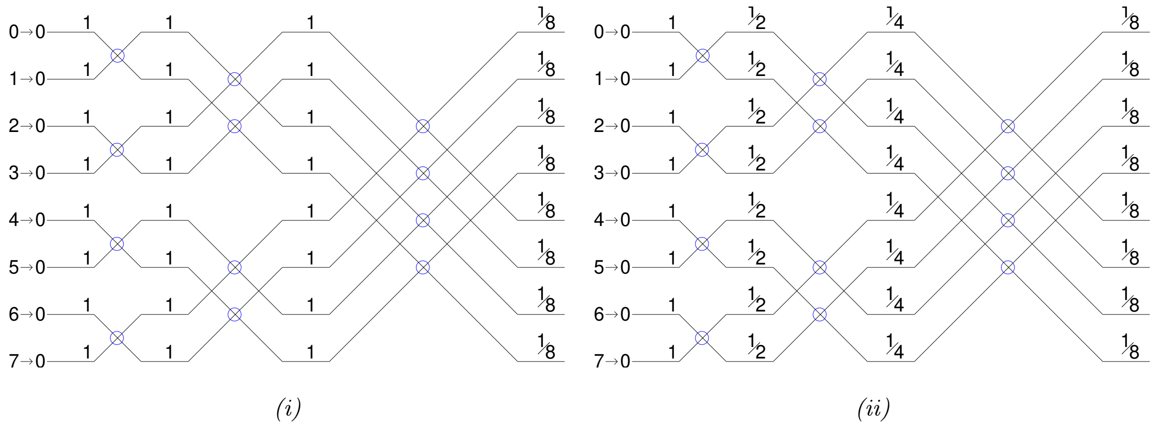

Figure 6.2: *Butterfly for final output* 0 *when the output contraint is enforced (i) at the last stage of the distribution Banyan and (ii) within the stages of the distribution Banyan. The numbers above the lines denote maximum available rate.*

harmful for the performance of the fabric. This cost-performance trade-off needs to be further studied for the actual Benes fabric.

In this report, we focus on per-output merging combined with hop-by-hop credit-based flow control. The mechanism of hop-by-hop credit-based flow control does not allow enforcing the desired ratios between flows destined to the same output link within the routing Banyan, this is illustrated in the following example. Suppose that flows $0 \to 0$ and $1 \to 0$ are the only active flows in the fabric and both send at 100%. The two flows meet at the `Merge` within stage 0 of the distribution Banyan and at the `SchRout` within stage 0 of the routing Banyan. However, each time `SchRout` serves a cell from one of the two flows, a credit for flow $01 \to 0$ reaches the first stage of the fabric. That is, the information regarding which flow was served is lost, and there is no mechanism for the `SchRout` to enforce the desired ratio between the service to flow $0 \to 0$ and the service to flow $1 \to 0$. If we did not use flow merging, then the `SchRout` would be able to enforce the desired ratios.

6.3 Input Constraints and Input Contention

We now turn to input constraint enforcement and input contention resolution. The input ports of the fabric enforce the inputs constraints and perform input contention resolution. However, input ports are not the only points in the fabric where input contention is resolved. Consider the following scenario: flows $0 \to 0$, $0 \to 1$, $1 \to 2$ and $7 \to 0$, $6 \to 1$, $5 \to 2$ are the only active flows in the fabric and all are persistent. Also assume that flows $7 \to 0$, $6 \to 1$ and $5 \to 2$ have the higher priority in the fabric. In this case, flows $0 \to 0$, $0 \to 1$ and $1 \to 2$ do not receive any service at outputs 0, 1 and 2, respectively, and backlog is created for flows $01 \to 0$, $01 \to 1$ and $01 \to 2$ at the output of the distribution switching element at stage 0. If flows $7 \to 0$, $6 \to 1$ and $5 \to 2$ stop sending, then input contention appears between flows $01 \to 0$, $01 \to 1$ and $01 \to 2$ for access to the link between the first and second stages of the distribution Banyan. The `SchDistr` at the head of the queues for flows $01 \rightarrow 0$, $01 \rightarrow 1$ and $01 \rightarrow 2$ is responsible for resolving that form of contention.

In general, input contention appears not only at the input ports of the fabric but also between the stages of the distribution Banyan and gets resolved by the `SchDistr`'s at the outputs of the distribution switching elements. The reason being that the fabric contains a total of $\log_P N$ distribution buffer stages, and the bandwidth between neigbouring buffer stages is fixed and equal to N cells per cell time. Note that the flow groups that meet at a given `SchDistr` originate from the same set of fabric inputs.

6.4 Summary

To sum up, input constraints are enforced at the input ports, while output constraints can be enforced either at the output ports, or the last stage of the distribution Banyan, or within the stages of the distribution Banyan. There exists a tree of `Merge`'s per fabric output which resolves output contention for that output in a hierarchical fashion. Input contention appears at the input ports and between the stages of the distribution Baynan and is resolved by the `SchDistr`'s.

In the actual benes fabric, there is an additional cause of internal delays, namely, traffic quantization and the distributed fashion of cell distribution. The scheduling architecture should be able to provide delay and bandwidth guarantees; the transient backlog caused by inefficiencies in the cell distribution method should not result in a failure to provide such guarantees. The scheduling architecture in the benes fabric is an important topic of future work.

Chapter 7

Related Work

Lucent's ATLANTA chip set uses a 3-stage buffered switching fabric with internal backpressure [27]. The middle stage of the ATLANTA chip set consists of multiple $\frac{N}{P} \times \frac{N}{P}$ bufferless crossbars, where P is the number of port interfaces connected to each input module. The ATLANTA chip set also uses per-output flow merging and cell distribution, but avoids resequencing because the middle stage is *bufferless*, thus, it does not reorder cells. However, it is not clear how to scale to larger port numbers and larger port rates and still use only three stages. Several other commercial chip sets also use backpressure in the ingress-switch-egress connection chain [13] [28].

A different approach to buffer placement in multistage switching fabrics is the *Parallel Packet Switch (PPS)* [10] [23], which is a derivative of central (shared) buffering and uses memory interleaving to provide scalable memory throughput. PPS is defined as a 3-stage fabric, where the bulk of the buffer space resides in the central stage. If one were to scale PPS to arbitrarily large aggregate throughput, one would end up with something along the lines of [29] : an input switching fabric (e.g. crossbar or banyan), which connects the input ports to the memory banks in the central stage and performs inverse multiplexing; the interleaved memory shared buffer in the central stage; and an output switching fabric (e.g. crossbar or banyan), which connects the memory banks to the output ports. This construction is similar to the construction of the Benes network from two banyans, which we review in section 2.1.1, but differs in buffer placement. The hard part of PPS is how to coordinate the traffic through its two fabrics so that the bulk of the queues end-up in the central stage, without using impractical centralized scheduling; several different proposals [15] and improvements [23] [29] exist. Their drawbacks are that either they need $O(N^2)$ logical queues in *each* of the $O(N)$ memory modules (banks), or they use a static schedule of length $O(N)$ in their switching fabric(s), or both. A quadratic number of queues poses serious cost scalability problems, while inverse multiplexing with a static schedule introduces long delays. By comparison, the architecture of this paper *(a)* features the same $O(N \cdot logN)$ number of switching elements and the same $O(N)$ number of memory modules; while *(b)* it never uses more than $O(N)$ queues per memory module or switching element; and *(c)* it uses dynamic scheduling in the switching fabric, so as to achieve short delays. Backpressure is the mechanism to coordinate the distributed operation of dynamic scheduling.

Chapter 8

Conclusions and Future Work

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 using the Benes network, the lowest-cost switching fabric that is free of internal blocking. Large buffer memories are only needed at the inputs of the system, to implement virtual output queues (VOQ); their number scales linearly with system size, the number of queues in each memory also scales linearly, while their throughput stays fixed. Internal backpressure is used in the Benes fabric, in order to provide: *(a)* low cost switching elements, since they only need on-chip buffer memory; *(b)* zero cell loss in the switching fabric, although buffer memories are small; *(c)* low system cost, since the fabric needs no internal speedup; *(d)* low system cost, since the fabric does not need redundant paths to handle cell conflicts using deflection routing; *(e)* low system cost, since no global scheduler is needed, and all scheduling and coordination is distributed; and *(f)* high system performance and high quality of service, even though system cost is kept low as detailed above.

To achieve all these, we had to extend the known per-flow backpressure architecture so as to make it applicable to multipath routing (inverse multiplexing) and cell resequencing, while keeping its cost manageable. We achieved this using an appropriate flow merging scheme that keeps the cost of backpressure down to $O(N)$ per switching element. We proved freedom from deadlock for a wide class of multipath cell distribution algorithms. Finally, using a cell-time-accurate simulator, *(a)* we showed that per-stage resequencing is preferable; *(b)* we found that cell distribution based on imbalance counts 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 25 to 50 % higher than ideal output queueing; and *(d)* 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.

The current paper describes the system architecture, the rationale behind it and how to make the system work. We plan to work on two areas on evolving the architecture: practical considerations and analytical modeling. With regard to practical considerations, we plan to work on *(a)* optimizations for 3-stage fabrics, and *(b)* simpler logical buffer organizations for the switching elements. While we have done some work on characterizing analytically the effects of cell distribution, this work needs to be extented, thus, we plan to *(a)* perform an analysis using network calculus that takes into account window flow control [30], and *(b)* investigate cell distribution methods based on theoretical results [31] [32]. Lastly, the scheduling architecture and multicasting support are important topics of future work.

Bibliography

[1] M. Marcus, "The Theory of Connecting Networks and their Complexity: a Review," *IEEE Proceedings*, vol. 65, no. 9, pp. 1263–1271, Sept. 1977.

[2] C.-L. Wu and T.-Y. Feng, "On a Class of Multistage Interconnection Networks," *IEEE Trans. on Computers*, vol. 29, no. 8, pp. 694–702, Aug. 1980.

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

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

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

[6] 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.

[7] 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, http://archvlsi.ics.forth.gr/atlasI/hoti98/.

[8] 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, http://www.bell-labs.com/org/113480/Papers/fabio-globecom98B.ps.

[9] 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, http://www.bell-labs.com/org/113480/Papers/dkh-globecom99.ps.

[10] Sundar Iyer, Amr A. Awadallah, and Nick McKeown, "Analysis of a Packet Switch with Memories Running Slower than the Line-Rate," in *IEEE INFOCOM*, Mar. 2000, http://klamath.stanford.edu/~sundaes/Papers/infocom2000.pdf.

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

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

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

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

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

[16] William J. Dally, "Virtual Channel Flow Control," *IEEE Transactions on Parallel and Distributed Systems*, vol. 3, no. 2, pp. 194–205, Mar. 1992.

[17] C. Ozveren, R. Simcoe, and G. Varghese, "Reliable and Efficient Hop-by-Hop Flow Control," *IEEE Journal on Selected Areas in Communication*, vol. 13, no. 4, pp. 642–650, May 1995.

[18] Quantum Flow Control Alliance, "Quantum Flow Control: A cell-relay protocol supporting an Available Bit Rate Service," July 1995, version 2.0.

[19] M. Katevenis, D. Serpanos, and E. Spyridakis, "Credit-Flow-Controlled ATM for MP Interconnection: the ATLAS I Single-Chip ATM Switch," in *HPCA*, Feb. 1998, pp. 47–56, http://archvlsi.ics.forth.gr/atlasI/atlasI_hpca98.ps.gz.

[20] M. Katevenis, "Fast Switching and Fair Control of Congested Flow in Broad-Band Networks," *IEEE Journal on Selected Areas in Communication*, vol. 5, no. 8, pp. 1315–1326, Oct. 1987.

[21] Manolis Katevenis, "Wide Links: $2 \cdot logN$-Stage Non-Blocking Networks with In-Order Packet Delivery," Internal Note, June 1995.

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

[23] D. A. Khotimsky and S. Krishnan, "Stability Analysis of a Parallel Packet Switch with Bufferless Input Demultiplexors," in *ICC 2001*, June 2001.

[24] Cheng-Shang Chang, Duan-Shin Lee, and Yi-Shean Jou, "Load Balanced Birkhoff-von Neumann Switches, Part II: Multi-stage Buffering," *Computer Communications, special issue on "Current Issues in Terabit Switching"*, 2001.

[25] D. A. Khotimsky and S. Krishnan, "Towards the Recognition of Parallel Packet Switches," in *GBN Workshop*, Apr. 2001, http://www.cs.columbia.edu/~sk/research/papers/pps-gbn01.html.

[26] Donpaul Stephens, "Implementing Distributed Packet Fair Queueing in a Scalable Switch Architecture," M.S. thesis, Carnegie Mellon University, May 1998.

[27] 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.

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

[29] C.-S. Chang, D.-S. Lee, and Y.-S. Jou, "Load Balanced Birkhoff-von Neumann Switches, Part I: One-stage Buffering," *IEEE HPSR Conf.*, May 2001, http://www.ee.nthu.edu.tw/cschang/PartI.ps.

[30] R. Agrawal, R. L. Cruz, C. Okino, and R. Rajan, "Performance Bounds for Flow Control Protocols," *IEEE/ACM Transactions on Networking*, vol. 7, no. 3, pp. 310–323, June 1999.

[31] Allan Borodin, Jon Kleinberg, Prabhakar Raghavan, Madhu Sudan, and David P. Williamson, "Adversarial Queueing Theory," in *Proc.* 28^{th} *ACM STOC*, May 1996, pp. 376–385.

[32] A. Richa M. Mitzenmacher and R. Sitaraman, "The Power of Two Random Choices: A survey of the Techniques and Results," in *Handbook of Randomized Computing, P. Pardalos, S. Rajasekaran, and J. Rolim, Eds.* Kluwer, 2000.

[33] Cyriel Minkenberg, *On Packet Switch Design*, Ph.D. thesis, Eindhoven University of Technology / IBM ZRL, Sept. 2001.

Appendix A

Proofs for chapter 5

In this chapter we use the following lemmas:

Lemma 1 *For a queue served by a FIFO scheduler of rate r and for $(0 \leq s \leq t)$, it holds:*

$$A_j(t-d) - A_j(s-d) \quad \leq \quad q_j(t-d) + r \cdot (t-s) \tag{A.1}$$

Proof: Let:

$$p_j(u,v) = A_j(v) - A_j(u) - r \cdot (v-u)$$

We consider the following cases:

- $(s \leq t < d) \iff (s-d \leq t-d < 0)$
  In this case, it holds:

$$\begin{aligned} p_j(s-d,t-d) &= A_j(t-d) - A_j(s-d) - r \cdot ((t-d)-(s-d)) \\ &= 0 - 0 - r \cdot (t-s) \\ &= -r \cdot (t-s) \\ &\leq 0 \Rightarrow \\ p_j(s-d,t-d) &\leq q_j(t-d) \end{aligned}$$

- $(d \leq t) \iff (0 \leq t-d)$
  We consider two subcases:
  - $(d \leq s \leq t) \iff (0 \leq s-d \leq t-d)$
    In this case, it holds:

$$\begin{aligned} q_j(u) &= \max_{0 \leq v \leq u} p_j(v,u) \Rightarrow \\ q_j(u) &\geq p_j(v,u),\ 0 \leq v \leq u \Rightarrow \\ q_j(t-d) &\geq p_j(s-d,t-d),\ 0 \leq s-d \leq t-d \end{aligned}$$

– $(0 \leq s < d \leq t) \iff (-d \leq s - d < 0 \leq t - d)$

In this case, it holds:

$$\begin{aligned}
p_j(s-d, t-d) &= A_j(t-d) - A_j(s-d) - r \cdot ((t-d) - (s-d)) \\
&= A_j(t-d) - A_j(s-d) - r \cdot (t-s) \\
&= A_j(t-d) - 0 - r \cdot (t-s) \\
&= A_j(t-d) - r \cdot t + r \cdot s \\
&< A_j(t-d) - r \cdot t + r \cdot d \\
&= A_j(t-d) - r \cdot (t-d) \\
&= p_j(0, t-d) \Rightarrow \\
p_j(s-d, t-d) &< p_j(0, t-d) \Rightarrow \\
p_j(s-d, t-d) &< q_j(t-d)
\end{aligned}$$

since, from subcase $(0 \leq s - d \leq t - d)$ for $s = d$ we get:

$$q_j(t-d) \quad \geq \quad p_j(0, t-d), \ 0 \leq t - d$$

Thus, in all cases it holds:

$$\begin{aligned}
q_j(t-d) &\geq p_j(s-d, t-d) \\
&\geq A_j(t-d) - A_j(s-d) - r \cdot ((t-d) - (s-d)) \\
&\geq A_j(t-d) - A_j(s-d) - r \cdot (t-s) \Rightarrow \\
A_j(t-d) - A_j(s-d) &\leq q_j(t-d) + r \cdot (t-s)
\end{aligned}$$

◁

Lemma 2

$$\begin{aligned}
\lceil a \rceil - \lceil b \rceil &\leq \lceil a - b \rceil \\
\lceil a \rceil + \lceil b \rceil &\leq \lceil a + b \rceil + 1 \\
\lceil a \rceil &< a + 1 \\
a - 1 &< \lfloor a \rfloor \\
\sum_{j=0}^{N-1} A_{i,j}(t) &\leq 1 \cdot t
\end{aligned}$$

Proof: Well known. ◁

A.1 Proof of Theorem 3

We will prove by induction that:

$$q_{i_k,p_k}^{D(k)}(t) \quad < \quad k \cdot N \tag{A.2}$$

$$d^{D(k)} \quad < \quad k \cdot N \tag{A.3}$$

We start with the base case, i.e. for $k = 1$:

$$\begin{aligned}
A^{D(1)}_{i_1,p_1}(t) - A^{D(1)}_{i_1,p_1}(s) &= \sum_{j=0}^{N-1}(A^{D(1)}_{i_1,p_1,j}(t) - A^{D(1)}_{i_1,p_1,j}(s)) \\
&\leq \sum_{j=0}^{N-1}(\lceil \frac{\sum_i A_{i,j}(t)}{P} \rceil - \lceil \frac{\sum_i A_{i,j}(s)}{P} \rceil) \\
&\leq \sum_{j=0}^{N-1} \lceil \frac{\sum_i (A_{i,j}(t) - A_{i,j}(s))}{P} \rceil \\
&\leq \lceil \frac{\sum_{j=0}^{N-1} \sum_i (A_{i,j}(t) - A_{i,j}(s))}{P} \rceil + (N-1) \\
&= \lceil \frac{\sum_i \sum_{j=0}^{N-1} (A_{i,j}(t) - A_{i,j}(s))}{P} \rceil + (N-1) \\
&\leq \lceil \frac{\sum_i (t-s)}{P} \rceil + (N-1) \\
&\leq \lceil \frac{P \cdot (t-s)}{P} \rceil + (N-1) \\
&= (t-s) + (N-1)
\end{aligned}$$

$$\begin{aligned}
q^{D(1)}_{i_1,p_1}(t) &= \max_{0 \leq s \leq t}[A^{D(1)}_{i_1,p_1}(t) - A^{D(1)}_{i_1,p_1}(s) - 1 \cdot (t-s)] \\
&\leq \max_{0 \leq s \leq t}[(t-s) + (N-1) - (t-s)] \\
&= \max_{0 \leq s \leq t}[(N-1)] \\
&< N
\end{aligned}$$

The relation $d^{D(1)} < 1 \cdot N$ follows from $q^{D(1)}_{i_1,p_1}(t) < 1 \cdot N$ and the fact that the queue is served by a work-conserving FIFO scheduler with a constant rate of 1. That is, $d^{D(1)}$ is bounded from above by N.

We now turn to the inductive step. Assume that $q^{D(k)}_{i_k,p_k}(t) < k \cdot N$, we will prove that $q^{D(k+1)}_{i_{k+1},p_{k+1}}(t) < (k+1) \cdot N$ also holds in this case. Since $q^{D(k)}_{i_k,p_k}(t) < k \cdot N$, it also holds that $d^{D(k)} < k \cdot N$, i.e. we also assume that $d^{D(k)}$ is bounded.

$$\begin{aligned}
A^{D(k+1)}_{i_{k+1},p_{k+1}}(t) - A^{D(k+1)}_{i_{k+1},p_{k+1}}(s) &= \sum_{j=0}^{N-1}(A^{D(k+1)}_{i_{k+1},p_{k+1},j}(t) - A^{D(k+1)}_{i_{k+1},p_{k+1},j}(s)) \\
&\leq \sum_{j=0}^{N-1}(\lceil \frac{\sum_{i_k} A^{D(k)}_{i_k,p_k,j}(t - d^{D(k)})}{P} \rceil - \lceil \frac{\sum_{i_k} A^{D(k)}_{i_k,p_k,j}(s - d^{D(k)})}{P} \rceil) \\
&\leq \sum_{j=0}^{N-1} \lceil \frac{\sum_{i_k} (A^{D(k)}_{i_k,p_k,j}(t - d^{D(k)}) - A^{D(k)}_{i_k,p_k,j}(s - d^{D(k)}))}{P} \rceil \\
&\leq \lceil \frac{\sum_{j=0}^{N-1} \sum_{i_k} (A^{D(k)}_{i_k,p_k,j}(t - d^{D(k)}) - A^{D(k)}_{i_k,p_k,j}(s - d^{D(k)}))}{P} \rceil + (N-1) \\
&= \lceil \frac{\sum_{i_k} \sum_{j=0}^{N-1} (A^{D(k)}_{i_k,p_k,j}(t - d^{D(k)}) - A^{D(k)}_{i_k,p_k,j}(s - d^{D(k)}))}{P} \rceil + (N-1) \\
&= \lceil \frac{\sum_{i_k} (A^{D(k)}_{i_k,p_k}(t - d^{D(k)}) - A^{D(k)}_{i_k,p_k}(s - d^{D(k)}))}{P} \rceil + (N-1)
\end{aligned}$$

$$
\begin{aligned}
&\leq \frac{\sum_{i_k}(A^{D(k)}_{i_k,p_k}(t-d^{D(k)}) - A^{D(k)}_{i_k,p_k}(s-d^{D(k)}))}{P} + 1 + (N-1) \\
&= \frac{1}{P}\cdot\sum_{i_k}(A^{D(k)}_{i_k,p_k}(t-d^{D(k)}) - A^{D(k)}_{i_k,p_k}(s-d^{D(k)})) + N
\end{aligned}
$$

$$
\begin{aligned}
q^{D(k+1)}_{i_{k+1},p_{k+1}}(t) &= \max_{0\leq s\leq t}[A^{D(k+1)}_{i_{k+1},p_{k+1}}(t) - A^{D(k+1)}_{i_{k+1},p_{k+1}}(s) - 1\cdot(t-s)] \\
&\leq \max_{0\leq s\leq t}[\frac{1}{P}\cdot\sum_{i_k}(A^{D(k)}_{i_k,p_k}(t-d^{D(k)}) - A^{D(k)}_{i_k,p_k}(s-d^{D(k)})) + N - (t-s)] \\
&= \frac{1}{P}\cdot\max_{0\leq s\leq t}[\sum_{i_k}(A^{D(k)}_{i_k,p_k}(t-d^{D(k)}) - A^{D(k)}_{i_k,p_k}(s-d^{D(k)})) - P\cdot(t-s)] + N \\
&= \frac{1}{P}\cdot\max_{0\leq s\leq t}[\sum_{i_k}(A^{D(k)}_{i_k,p_k}(t-d^{D(k)}) - A^{D(k)}_{i_k,p_k}(s-d^{D(k)}) - (t-s))] + N \\
&\leq \frac{1}{P}\cdot\sum_{i_k}\max_{0\leq s\leq t}[A^{D(k)}_{i_k,p_k}(t-d^{D(k)}) - A^{D(k)}_{i_k,p_k}(s-d^{D(k)}) - (t-s)] + N \\
&\leq \frac{1}{P}\cdot\sum_{i_k}\max_{0\leq s\leq t}[q^{D(k)}_{i_k,p_k}(t-d^{D(k)}) + (t-s) - (t-s)] + N \\
&= \frac{1}{P}\cdot\sum_{i_k}\max_{0\leq s\leq t}[q^{D(k)}_{i_k,p_k}(t-d^{D(k)})] + N \\
&= \frac{1}{P}\cdot\sum_{i_k}q^{D(k)}_{i_k,p_k}(t-d^{D(k)}) + N \Rightarrow \\
q^{D(k+1)}_{i_{k+1},p_{k+1}}(t) &< \frac{1}{P}\cdot\sum_{i_k}(k\cdot N) + N \\
&= \frac{1}{P}\cdot k\cdot N\cdot\sum_{i_k}1 + N \\
&= \frac{1}{P}\cdot k\cdot N\cdot P + N \\
&= (k+1)\cdot N
\end{aligned}
$$

The relation $d^{D(k+1)} < (k+1)\cdot N$ follows from $q^{D(k+1)}_{i_{k+1},p_{k+1}}(t) < (k+1)\cdot N$ and the fact that the queue is served by a work-conserving FIFO scheduler with a constant rate of 1.

This completes the proof of eq. A.2 and eq. A.3.

A.2 Proof of Theorem 4

We will first prove by induction the following lemma:

Lemma 3

$$
A^{D(k)}_{i_k,p_k,j}(t) - A^{D(k)}_{i_k,p_k,j}(s) \quad\leq\quad k + \frac{1}{P^k}\cdot\sum_{i_{k-1}}\cdots\sum_{i}(A_{i,j}(t-\sum_{s=1}^{k-1}d^{D(s)}) - A_{i,j}(s-\sum_{s=1}^{k-1}d^{D(s)})) \quad (A.4)
$$

In the above equation, i_{k-1} is a function of i_k and runs over P switches for each value of i_k, and so on, so forth, down to i_1 which is a function of i_2 and runs over P switches for each value of i_2 and i which is a function of i_1 and runs over P input links for each value of i_1.

Proof: It holds:

$$\begin{aligned}
A^{D(1)}_{i_1,p_1,j}(t) - A^{D(1)}_{i_1,p_1,j}(s) &\leq \lceil\frac{\sum_i A_{i,j}(t)}{P}\rceil - \lceil\frac{\sum_i A_{i,j}(s)}{P}\rceil \\
&\leq \lceil\frac{\sum_i (A_{i,j}(t) - A_{i,j}(s))}{P}\rceil \\
&\leq \frac{\sum_i (A_{i,j}(t) - A_{i,j}(s))}{P} + 1 \\
&= \frac{1}{P}\cdot\sum_i (A_{i,j}(t) - A_{i,j}(s)) + 1
\end{aligned}$$

$$\begin{aligned}
A^{D(k+1)}_{i_{k+1},p_{k+1},j}(t) - A^{D(k+1)}_{i_{k+1},p_{k+1},j}(s) &\leq \lceil\frac{\sum_{i_k} A^{D(k)}_{i_k,p_k,j}(t-d^{D(k)})}{P}\rceil - \lceil\frac{\sum_{i_k} A^{D(k)}_{i_k,p_k,j}(s-d^{D(k)})}{P}\rceil \\
&\leq \lceil\frac{\sum_{i_k} (A^{D(k)}_{i_k,p_k,j}(t-d^{D(k)}) - A^{D(k)}_{i_k,p_k,j}(s-d^{D(k)}))}{P}\rceil \\
&\leq \frac{\sum_{i_k} (A^{D(k)}_{i_k,p_k,j}(t-d^{D(k)}) - A^{D(k)}_{i_k,p_k,j}(s-d^{D(k)}))}{P} + 1 \\
&\leq \frac{1}{P}\cdot\sum_{i_k} (A^{D(k)}_{i_k,p_k,j}(t-d^{D(k)}) - A^{D(k)}_{i_k,p_k,j}(s-d^{D(k)})) + 1
\end{aligned}$$

That is:

$$A^{D(1)}_{i_1,p_1,j}(t) - A^{D(1)}_{i_1,p_1,j}(s) \leq \frac{1}{P}\cdot\sum_i (A_{i,j}(t) - A_{i,j}(s)) + 1 \tag{A.5}$$

$$A^{D(k+1)}_{i_{k+1},p_{k+1},j}(t) - A^{D(k+1)}_{i_{k+1},p_{k+1},j}(s) \leq \frac{1}{P}\cdot\sum_{i_k} (A^{D(k)}_{i_k,p_k,j}(t-d^{D(k)}) - A^{D(k)}_{i_k,p_k,j}(s-d^{D(k)})) + 1 \tag{A.6}$$

Eq. A.4 holds for $k = 1$ since it reduces to eq. A.5. Assume that eq. A.4 holds for some value k. Then, starting with eq. A.6, we will show that it also holds for value $k + 1$:

$$\begin{aligned}
&A^{D(k+1)}_{i_{k+1},p_{k+1},j}(t) - A^{D(k+1)}_{i_{k+1},p_{k+1},j}(s) = \\
&\leq 1 + \frac{1}{P}\cdot\sum_{i_k}(A^{D(k)}_{i_k,p_k,j}(t-d^{D(k)}) - A^{D(k)}_{i_k,p_k,j}(s-d^{D(k)})) \\
&\leq 1 + \frac{1}{P}\cdot\sum_{i_k}(k + \frac{1}{P^k}\cdot\sum_{i_{k-1}}\dots\sum_i(A_{i,j}(t-d^{D(k)}-\sum_{s=1}^{k-1}d^{D(s)}) - A_{i,j}(s-d^{D(k)}-\sum_{s=1}^{k-1}d^{D(s)})) \\
&= 1 + \frac{1}{P}\cdot\sum_{i_k}k + \frac{1}{P}\cdot\frac{1}{P^k}\cdot\sum_{i_k}\sum_{i_{k-1}}\dots\sum_i(A_{i,j}(t-\sum_{s=1}^{k}d^{D(s)}) - A_{i,j}(s-\sum_{s=1}^{k}d^{D(s)})) \\
&= (1+k) + \frac{1}{P^{k+1}}\cdot\sum_{i_k}\dots\sum_i(A_{i,j}(t-\sum_{s=1}^{k}d^{D(s)}) - A_{i,j}(s-\sum_{s=1}^{k}d^{D(s)}))
\end{aligned}$$

since:

$$\frac{1}{P}\cdot\sum_{i_k}k = \frac{1}{P}\cdot k\cdot\sum_{i_k}1 = \frac{1}{P}\cdot k\cdot P = k$$

This completes the proof of eq. A.4. ◁

Remember that we have set $L = \log_P N$ and $d^D = \sum_{s=1}^{L-1} d^{D(s)}$. By instantiating eq. A.4 for $k = L$, and for $(0 \leq s \leq t)$, we have:

$$A^{D(L)}_{i_L,p_L,j}(t) - A^{D(L)}_{i_L,p_L,j}(s) \leq L + \frac{1}{P^L}\cdot\sum_{i_{L-1}}\dots\sum_i(A_{i,j}(t-d^D) - A_{i,j}(s-d^D))$$

$$
\begin{aligned}
&= L + \frac{1}{N} \cdot \sum_{i=0}^{N-1} (A_{i,j}(t-d^D) - A_{i,j}(s-d^D)) \\
&= L + \frac{1}{N} \cdot (A_j(t-d^D) - A_j(s-d^D)) \\
&\leq L + \frac{1}{N} \cdot (q_j(t-d^D) + (t-s)) \\
&= L + \frac{q_j(t-d^D)}{N} + \frac{t-s}{N}
\end{aligned}
$$

For the schedulers that feed the middle links, it holds:

$$
C^{D(L)}_{i_L,p_L,j}(t) - C^{D(L)}_{i_L,p_L,j}(s) \quad = \quad \lfloor \frac{t-s}{N} \rfloor \quad > \quad \frac{t-s}{N} - 1
$$

Thus:

$$
\begin{aligned}
q^{D(L)}_{i_L,p_L,j}(t) &= \max_{0\leq s\leq t} [A^{D(L)}_{i_L,p_L,j}(t) - A^{D(L)}_{i_L,p_L,j}(s) - (C^{D(L)}_{i_L,p_L,j}(t) - C^{D(L)}_{i_L,p_L,j}(s))] \\
&< \max_{0\leq s\leq t} [L + \frac{q_j(t-d^D)}{N} + \frac{t-s}{N} - (\frac{t-s}{N} - 1)] \\
&= \max_{0\leq s\leq t} [L + 1 + \frac{q_j(t-d^D)}{N}] \\
&= \frac{q_j(t-d^D)}{N} + L + 1
\end{aligned}
$$

This completes the proof of eq. 5.7.

We now turn to eq. 5.8. Let $d = q_j(t-d^D) + d^M$, also remember that we have set $d^M = (L+2) \cdot N$. It holds:

$B^{D(L)}_{i_L,p_L,j}(t+d) - A^{D(L)}_{i_L,p_L,j}(t) =$

$$
\begin{aligned}
&= \min_{0\leq s\leq t+d} [A^{D(L)}_{i_L,p_L,j}(s) + C^{D(L)}_{i_L,p_L,j}(t+d) - C^{D(L)}_{i_L,p_L,j}(s)] - A^{D(L)}_{i_L,p_L,j}(t) \\
&= \min_{0\leq s\leq t+d} [(C^{D(L)}_{i_L,p_L,j}(t+d) - C^{D(L)}_{i_L,p_L,j}(s)) - (A^{D(L)}_{i_L,p_L,j}(t) - A^{D(L)}_{i_L,p_L,j}(s))] \\
&= \min \begin{bmatrix} \min_{0\leq s\leq t} & [(C^{D(L)}_{i_L,p_L,j}(t+d) - C^{D(L)}_{i_L,p_L,j}(s)) - (A^{D(L)}_{i_L,p_L,j}(t) - A^{D(L)}_{i_L,p_L,j}(s))], \\ \min_{t<s\leq t+d} & [(C^{D(L)}_{i_L,p_L,j}(t+d) - C^{D(L)}_{i_L,p_L,j}(s)) - (A^{D(L)}_{i_L,p_L,j}(t) - A^{D(L)}_{i_L,p_L,j}(s))] \end{bmatrix}
\end{aligned}
$$

For $(0 \leq s \leq t)$, it holds:

$$
\begin{aligned}
C^{D(L)}_{i_L,p_L,j}(t+d) - C^{D(L)}_{i_L,p_L,j}(s) &> \frac{t+d-s}{N} - 1 \\
A^{D(L)}_{i_L,p_L,j}(t) - A^{D(L)}_{i_L,p_L,j}(s) &\leq \frac{q_j(t-d^D)}{N} + L + 1 \\
&\leq \frac{d-d^M}{N} + L + 1
\end{aligned}
$$

For $(t < s \leq t+d)$, it holds:

$$
\begin{aligned}
C^{D(L)}_{i_L,p_L,j}(t+d) - C^{D(L)}_{i_L,p_L,j}(s) &\geq 0 \\
A^{D(L)}_{i_L,p_L,j}(t) - A^{D(L)}_{i_L,p_L,j}(s) &\leq 0
\end{aligned}
$$

Thus:

$$
\begin{aligned}
B^{D(L)}_{i_L,p_L,j}(t+d) - A^{D(L)}_{i_L,p_L,j}(t) &\geq \min[\min_{0\leq s\leq t}[\frac{t+d-s}{N} - 1 - \frac{d-d^M}{N} - L - 1], 0+0] \\
&\geq \min[\min_{0\leq s\leq t}[\frac{t-s}{N} + \frac{d^M}{N} - L - 2], 0] \\
&\geq \min[\min_{0\leq s\leq t}[\frac{t-s}{N} + \frac{(L+2)\cdot N}{N} - L - 2], 0] \\
&\geq \min[\min_{0\leq s\leq t}[\frac{t-s}{N}], 0] \\
&\geq \min[0, 0] \\
&\geq 0
\end{aligned}
$$

This completes the proof of eq. 5.8.

A.3 Proof of Theorem 5

In this section we use symbol $A(t)$ to denote both the number of cells that have arrived at some queue by time t and the set of cells themselves. Whether $A(t)$ refers to the set of the cells or the size of that set will be clear from the context. Similarly for $B(t)$.

A.3.1 Resequencing Buffers at stage $L-1$ of the Routing Banyan

Let $d_D(t) = t + q_j(t) + d^D + d^M$.

- As noted in Table 5.4, every cell that arrives at some fabric input by time t, will cross some middle link by time $d_D(t)$.

- Due to jitter control, a cell that arrives at some fabric input in a given time is succesively enqueued at the output buffers of the distribution switching elements *before* any cell that is destined to the same output and arrives at some fabric input in a following cell time. Moreover, cell distribution is performed on a per-flow basis and flows are formed using per-output flow merging. Thus, each cell need wait at the resequencing buffers only for cells that are destined to the same output and arrived at some fabric input in the same or previous cell time with that cell.

- It is $d_D(s) \leq d_D(t)$, $\forall$ $(s \leq t)$, thus, every cell that arrives at some fabric input before or at time t and is destined to output j reaches the resequencing buffer by time $d_D(t)$.

Thus, every cell that arrives at some fabric input by time t and is destined to output j will leave the resequencing buffer at stage $L-1$ of the routing Banyan by time $d_D(t)$.

In the shadow FIFO switch, a cell that arrives at some fabric input at time t and is destined to output j is served by the output scheduler between time $t + q_j(t) - (N-1)$ and $t + q_j(t)$ by definition of the FIFO scheduling discipline. That is, every cell passes through the resequencing buffer at stage $L-1$ of the routing Banyan delayed by less than $d^D + d^M + N$ with regard to the time it leaves the output of the shadow FIFO switch. Thus, the set of cells that have passed through the resequencing buffers at stage $L-1$ of the routing Banyan by time $t' + d^D + d^M + N$ is a *superset* of the cells that have been served by the shadow FIFO switch by time t'. Let $d^{mid} = d^D + d^M + N$, then it holds:

$$\bigcup_{i_L, q_L} B^{R(L),rsq}_{i_L,q_L,j}(t' + d^{mid}) \supseteq B_j(t')$$

Now, assume that the resequencing buffers at stage $L-1$ of the routing Banyan perform jitter control so that a cell that arrives at the fabric input at time t and is destined to output j leaves the resequencing buffer exactly at time $d_D(t)$. Then, the set of cells that have passed through the resequencing buffers at stage $L-1$ of the routing Banyan by time $t' + d^{mid}$ is *exactly* the set of the cells that have been served by the shadow FIFO switch by time t'. That is:

$$\bigcup_{i_L, q_L} B^{R(L),rsq}_{i_L,q_L,j}(t' + d^{mid}) \equiv B_j(t')$$

Note that for the resequencing buffers to implement this form of jitter control, they need to keep track of the shadow FIFO switch, which requires global knowledge. We make this assumption to simplify the analysis, we hope to eliminate it in a future work by bounding from above the set of cells served by the schedulers at the head of the middle-stage buffers.

The above form of jitter control implies that up to N cells may leave the resequencing buffers within a signle cell time, but also implies that: the set of cells destined to output j that have passed through the resequencing buffers at stage $L-1$ of the routing Banyan by time $t' + d^{mid}$ is *exactly* the set of the cells that are destined to output j and have arrived at all inputs by some time t, where t is common for all fabric inputs. That is:

$$\bigcup_{i_L, q_L} B^{R(L),rsq}_{i_L,q_L,j}(t' + d^{mid}) \quad \equiv \quad B_j(t') \; \equiv \; \bigcup_{i=0}^{N-1} A_{i,j}(t)$$

Thus, in order to compute $B^{R(L),rsq}_{i_L,q_L,j}(t' + d^{mid})$, $\forall \; (i_L, q_L)$, we need to study how $\bigcup_{i=0}^{N-1} A_{i,j}(t)$ is distributed over the links at the heart of the fabric. This is done in the next section.

A.3.2 Number of Cells through each Benes Subnetwork

We use the notation shown in Table A.1.

Remember that jitter control within the distribution Banyan prevents cells that arrive at the fabric in a given cell time and are destined to a given output of the fabric to overtake cells that are destined to the same output but arrived in previous cell times. That is, a cell that arrives at some input of the fabric in a given cell time and is destined to a given output gets distributed at each stage before any cell that arrives in a following cell time and is destined to the same output. Thus, the following equation holds:

$$\lfloor \frac{1}{P} \cdot \sum_{l_k \in \; inp(l_{k+1})} A_{j/l_k}(t) \rfloor \;\leq\; A_{j/l_{k+1}}(t) \;\leq\; \lceil \frac{1}{P} \cdot \sum_{l_k \in \; inp(l_{k+1})} A_{j/l_k}(t) \rceil$$

$A_{j/l_k}(t)$	number of cells that:	*have* arrived at some input by time t are destined for final output j *have been* or *will be* forwarded through link l_k
	l_k is the index of a link between stages $k-1$ and k of the distribution Banyan	
$A_{j/B_k}(t)$	number of cells that:	*have* arrived at some input by time t are destined for final output j *have been* or *will be* forwarded through Benes subnetwork B_k
	B_k identifies a $P^{L-k} \times P^{L-k}$ Benes subnetwork embedded within the $N \times N$ fabric	

Table A.1: *Notation for the variables that describe how the set of cells $\bigcup_{i=0}^{N-1} A_{i,j}(t)$ is distributed within the Benes fabric.*

In the above equation, $l_k \in \ inp(l_{k+1})$ means that l_k depends on l_{k+1} and runs over the P input links of the switching element of which the link with index l_{k+1} is an output link. Note that we need not time-shift $A_{j/l_k}(t)$ since, by definition, $A_{j/l_k}(t)$ includes all cells that *have been* or *will be* forwaded. With regard to $A_{j/B_k}(t)$, it holds:

$$A_{j/B_k}(t) \quad = \quad \sum_{l_k \in \ inp(B_k)} A_{j/l_k}(t)$$

In the above equation, $l_k \in \ inp(B_k)$ means that l_k runs over the $P^{L-k} = \frac{N}{P^k}$ input links of Benes subnetwork B_k.

Lemma 4 *For every Benes subnetworks B_k, $(0 \leq k \leq L = \log_P N)$, it holds:*

$$|A_{j/B_k}(t) - \frac{A_j(t)}{P^k}| \quad \leq \quad k \cdot \frac{N}{P^k} \tag{A.7}$$

Proof: For $k = 0$, eq. A.7 becomes:

$$|A_{j/B_0}(t) - \frac{A_j(t)}{P^0}| \ \leq \ 0 \cdot \frac{N}{P^0} \iff |A_{j/B_0}(t) - A_j(t)| \ \leq \ 0$$

which is true, since B_0 is the Benes fabric itself.

Assume that eq. A.7 holds for some value k, we will show that it also holds for value $k+1$ in that case. It holds:

$$\begin{aligned}
A_{j/B_{k+1}}(t) \quad &= \quad \sum_{l_{k+1} \in \ inp(B_{k+1})} A_{j/l_{k+1}}(t) \\
&\geq \quad \sum_{l_{k+1} \in \ inp(B_{k+1})} \lfloor \frac{1}{P} \cdot \sum_{l_k \in \ inp(l_{k+1})} A_{j/l_k}(t) \rfloor \\
&> \quad \sum_{l_{k+1} \in \ inp(B_{k+1})} (\frac{1}{P} \cdot \sum_{l_k \in \ inp(l_{k+1})} A_{j/l_k}(t) - 1) \\
&= \quad \frac{1}{P} \cdot \sum_{l_{k+1} \in \ inp(B_{k+1})} \ \sum_{l_k \in \ inp(l_{k+1})} A_{j/l_k}(t) - \sum_{l_{k+1} \in \ inp(B_{k+1})} 1
\end{aligned}$$

$$
\begin{aligned}
&= \frac{1}{P} \cdot \sum_{l_k \in\, inp(B_k)} A_{j/l_k}(t) - \frac{N}{P^{k+1}} \\
&= \frac{1}{P} \cdot A_{j/B_k}(t) - \frac{N}{P^{k+1}} \\
&\geq \frac{1}{P} \cdot (\frac{A_j(t)}{P^k} - k \cdot \frac{N}{P^k}) - \frac{N}{P^{k+1}} \\
&= \frac{A_j(t)}{P^{k+1}} - k \cdot \frac{N}{P^{k+1}} - \frac{N}{P^{k+1}} \\
&= \frac{A_j(t)}{P^{k+1}} - (k+1) \cdot \frac{N}{P^{k+1}} \Rightarrow \\
A_{j/B_{k+1}}(t) - \frac{A_j(t)}{P^{k+1}} &\geq -(k+1) \cdot \frac{N}{P^{k+1}}
\end{aligned}
$$

Similarly, we can prove that:

$$A_{j/B_{k+1}}(t) - \frac{A_j(t)}{P^{k+1}} \quad \leq \quad (k+1) \cdot \frac{N}{P^{k+1}}$$

Thus:

$$-(k+1) \cdot \frac{N}{P^{k+1}} \ \leq \ A_{j/B_{k+1}}(t) - \frac{A_j(t)}{P^{k+1}} \ \leq \ (k+1) \cdot \frac{N}{P^{k+1}} \iff$$

equivalently:

$$|A_{j/B_{k+1}}(t) - \frac{A_j(t)}{P^{k+1}}| \ \leq \ (k+1) \cdot \frac{N}{P^{k+1}}$$

This completes the proof of eq. A.7.

$\triangleleft$

A.3.3 Delay through the stages of the Routing Banyan

In this section, we prove by induction an upper bound on $d^{R(k)}$, $(L = \log_P N \geq k \geq 1)$. In the analysis we assume that the input queues of the routing switching elements perform jitter-control so that all cells face delay $d^{R(k)}$ from the time they enter the output queue of the switching element at stage $k-1$ of the routing Banyan until the time they enter the output queue of the switching element at the next stage. Effectively, we show that the set $B_j(t')$ is transfered from stage $k-1$ of the routing Banyan to stage k with delay $d^{R(k)}$.

We start with the base case, i.e. for $k = L = \log_P N$. From eq. A.7, for $k = L$ we get:

$$
\begin{aligned}
|A_{j/B_L}(t) - \frac{A_j(t)}{P^L}| &\leq L \cdot \frac{N}{P^L} \Rightarrow \\
|A_{j/B_L}(t) - \frac{A_j(t)}{N}| &\leq L
\end{aligned}
$$

Moreover, it holds:

$$B_j(t') \ = \ \bigcup_{i=0}^{N-1} A_{i,j}(t) \ = \ A_j(t)$$

and

$$B^{R(L),rsq}_{i_L,q_L,j}(t' + d^{mid}) \ = \ A_{j/l_L}(t) \ = \ A_{j/B_L}(t)$$

where l_L is the link connected to input port q_L of switch i_L, and B_L is the degenerate $P^{L-L} \times P^{L-L}$ i.e. 1×1 Benes fabric which corresponds to the middle link l_L. Thus:

$$|B^{R(L),rsq}_{i_L,q_L,j}(t' + d^{mid}) - \frac{B_j(t')}{N}| \quad \leq \quad L$$

For $(0 \leq s \leq t)$ it holds:

$$\begin{aligned}
A^{R(L)}_{i_L,p_L,j}(t) - A^{R(L)}_{i_L,p_L,j}(s) &= \sum_{q_L}(B^{R(L),rsq}_{i_L,q_L,j}(t) - B^{R(L),rsq}_{i_L,q_L,j}(s)) \\
&\leq \sum_{q_L}((\frac{B_j(t - d^{mid})}{N} + L) - (\frac{B_j(s - d^{mid})}{N} - L)) \\
&= \sum_{q_L}(\frac{B_j(t - d^{mid}) - B_j(s - d^{mid})}{N} + 2 \cdot L) \\
&= P \cdot (\frac{B_j(t - d^{mid}) - B_j(s - d^{mid})}{N} + 2 \cdot L) \\
&= \frac{P}{N} \cdot (B_j(t - d^{mid}) - B_j(s - d^{mid})) + 2 \cdot P \cdot L \\
&\leq \frac{P}{N} \cdot 1 \cdot ((t - d^{mid}) - (s - d^{mid})) + 2 \cdot P \cdot L \\
&= \frac{P}{N} \cdot (t - s) + 2 \cdot P \cdot L \\
C^{R(L)}_{i_L,p_L,j}(t) - C^{R(L)}_{i_L,p_L,j}(s) &= \lfloor \frac{t - s}{P^{L-1}} \rfloor \\
&> \frac{t - s}{P^{L-1}} - 1 \\
&= \frac{P}{N} \cdot (t - s) - 1
\end{aligned}$$

Thus:

$$\begin{aligned}
q^{R(L)}_{i_L,p_L,j}(t) &= \max_{0 \leq s \leq t}[(A^{R(L)}_{i_L,p_L,j}(t) - A^{R(L)}_{i_L,p_L,j}(s)) - (C^{R(L)}_{i_L,p_L,j}(t) - C^{R(L)}_{i_L,p_L,j}(s))] \\
&< \max_{0 \leq s \leq t}[(\frac{P}{N} \cdot (t - s) + 2 \cdot P \cdot L) - (\frac{P}{N} \cdot (t - s) - 1)] \\
&= \max_{0 \leq s \leq t}[2 \cdot P \cdot L + 1] \\
&= 2 \cdot P \cdot L + 1
\end{aligned}$$

We also set $d_B = 2 \cdot L \cdot N + N$, then it holds:

$$B^{R(L)}_{i_L,p_L,j}(t + d_B) - A^{R(L)}_{i_L,p_L,j}(t) =$$

$$\begin{aligned}
&= \min_{0 \leq s \leq t + d_B}[(A^{R(L)}_{i_L,p_L,j}(s) + (C^{R(L)}_{i_L,p_L,j}(t + d_B) - C^{R(L)}_{i_L,p_L,j}(s))] - A^{R(L)}_{i_L,p_L,j}(t) \\
&= \min_{0 \leq s \leq t + d_B}[(C^{R(L)}_{i_L,p_L,j}(t + d_B) - C^{R(L)}_{i_L,p_L,j}(s)) - (A^{R(L)}_{i_L,p_L,j}(t) - A^{R(L)}_{i_L,p_L,j}(s))] \\
&= \min \begin{bmatrix} \min_{0 \leq s \leq t} & [(C^{R(L)}_{i_L,p_L,j}(t + d) - C^{R(L)}_{i_L,p_L,j}(s)) - (A^{R(L)}_{i_L,p_L,j}(t) - A^{R(L)}_{i_L,p_L,j}(s))], \\ \min_{t < s \leq t + d_B} & [(C^{R(L)}_{i_L,p_L,j}(t + d) - C^{R(L)}_{i_L,p_L,j}(s)) - (A^{R(L)}_{i_L,p_L,j}(t) - A^{R(L)}_{i_L,p_L,j}(s))] \end{bmatrix}
\end{aligned}$$

For $(0 \leq s \leq t)$, it holds:

$$C^{R(L)}_{i_L,p_L,j}(t + d_B) - C^{R(L)}_{i_L,p_L,j}(s) \quad > \quad \frac{P}{N} \cdot (t + d_B - s) - 1$$

$$
\begin{aligned}
&= \frac{P}{N}\cdot(t-s)+\frac{P}{N}\cdot d_B-1 \\
A^{R(L)}_{i_L,p_L,j}(t)-A^{R(L)}_{i_L,p_L,j}(s) &\leq \frac{P}{N}\cdot(t-s)+2\cdot P\cdot L
\end{aligned}
$$

For $(t < s \leq t+d_B)$, it holds:

$$
\begin{aligned}
C^{R(L)}_{i_L,p_L,j}(t+d_B)-C^{R(L)}_{i_L,p_L,j}(s) &\geq 0 \\
A^{R(L)}_{i_L,p_L,j}(t)-A^{R(L)}_{i_L,p_L,j}(s) &\leq 0
\end{aligned}
$$

Thus:

$$
\begin{aligned}
&B^{R(L)}_{i_L,p_L,j}(t+d_B)-A^{R(L)}_{i_L,p_L,j}(t) \geq \\
&\quad\geq \min[\min_{0\leq s\leq t}[(\frac{P}{N}\cdot(t-s)+\frac{P}{N}\cdot d_B-1)-(\frac{P}{N}\cdot(t-s)+2\cdot P\cdot L)],0+0] \\
&\quad\geq \min[\min_{0\leq s\leq t}[\frac{P}{N}\cdot(d_B-2\cdot L\cdot N)-1],0] \\
&\quad\geq \min[\min_{0\leq s\leq t}[\frac{P}{N}\cdot N-1)],0] \\
&\quad\geq \min[\min_{0\leq s\leq t}[P-1],0] \\
&\quad\geq 0
\end{aligned}
$$

That is:

$$
d^{R(L)} \ \leq \ d_B \ = \ 2\cdot L\cdot N+N
$$

The above equation also implies that the set $B_j(t')$ passes through stage $L-1$ of the routing Banyan with a maximum delay of $d^{R(L)} \ \leq \ (2\cdot L+1)\cdot N$ cell times, i.e.

$$
\bigcup_{i_L,p_L} B^{R(L)}_{i_L,p_L,j}(t'+d^{mid}+d^{R(L)}) \ \supseteq \ B_j(t')
$$

We now turn to the inductive step. Let $d^k_R=\sum_{s=L}^{k} d^{R(s)}$. Assume that d^{k+1}_R is bounded and:

$$
\bigcup_{i_{k+1},p_{k+1}} B^{R(k+1)}_{i_{k+1},p_{k+1},j}(t'+d^{mid}+d^{k+1}_R) \ \ \supseteq \ \ B_j(t')
$$

We will show that d^k_R is also bounded and:

$$
\bigcup_{i_k,p_k} B^{R(k)}_{i_k,p_k,j}(t'+d^{mid}+d^{k}_R) \ \ \supseteq \ \ B_j(t')
$$

also holds in this case.

As explained in section A.3.1, a cell that arrives at time t at some fabric input and is destined to output j need wait at the resequencing buffers only for cells that are destined to output j and have already arrived at some fabric input by time t. That is, a cell that is served at the shadow FIFO switch at time t' need only wait for cells in the set $B_j(t')$. The above observation, along with the inductive hypothesis $\bigcup_{i_{k+1},p_{k+1}} B^{R(k+1)}_{i_{k+1},p_{k+1},j}(t'+d^{mid}+d^{k+1}_R) \ \supseteq \ B_j(t')$ and jitter control at the input queues of the routing switching elements imply that:

$$
\bigcup_{i_k,q_k} B^{R(k),rsq}_{i_k,q_k,j}(t'+d^{mid}+d^{k+1}_R) \ \ \equiv \ \ B_j(t')
$$

From eq. A.7, we have:

$$|A_{j/B_k}(t) - \frac{A_j(t)}{P^k}| \quad \leq \quad k \cdot \frac{N}{P^k}$$

Moreover, it holds:

$$B_j(t') \;=\; \bigcup_{i=0}^{N-1} A_{i,j}(t) \;=\; A_j(t)$$

and

$$\begin{aligned}\bigcup_{i_k,q_k} B^{R(k),rsq}_{i_k,q_k,j}(t' + d^{mid} + d^{k+1}_R) \quad &\equiv \quad B_j(t') \;\equiv\; \bigcup_{i=0}^{N-1} A_{i,j}(t) \Rightarrow \\ B^{R(k),rsq}_{i_k,q_k,j}(t' + d^{mid} + d^{k+1}_R) \quad &\equiv \quad A_{j/B_k}(t)\end{aligned}$$

where B_k is the $P^{L-k} \times P^{L-k}$ Benes subnetwork which is connected to input port q_k of switch i_k. Thus:

$$|B^{R(k),rsq}_{i_k,q_k,j}(t' + d^{mid} + d^{k+1}_R) - \frac{B_j(t')}{P^k}| \quad \leq \quad k \cdot \frac{N}{P^k}$$

For $(0 \leq s \leq t)$ it holds:

$$\begin{aligned}A^{R(k)}_{i_k,p_k,j}(t) - A^{R(k)}_{i_k,p_k,j}(s) \quad &= \quad \sum_{q_k}(B^{R(k),rsq}_{i_k,q_k,j}(t) - B^{R(k),rsq}_{i_k,q_k,j}(s)) \\ &\leq \quad \sum_{q_k}((\frac{B_j(t - d^{mid} - d^{k+1}_R)}{P^k} + k \cdot \frac{N}{P^k}) - (\frac{B_j(s - d^{mid} - d^{k+1}_R)}{P^k} - k \cdot \frac{N}{P^k})) \\ &= \quad \sum_{q_k}(\frac{B_j(t - d^{mid} - d^{k+1}_R) - B_j(s - d^{mid} - d^{k+1}_R)}{P^k} + 2 \cdot k \cdot \frac{N}{P^k}) \\ &= \quad \frac{P}{P^k} \cdot (B_j(t - d^{mid} - d^{k+1}_R) - B_j(s - d^{mid} - d^{k+1}_R) + 2 \cdot k \cdot N) \\ &\leq \quad \frac{1}{P^{k-1}} \cdot (1 \cdot ((t - d^{mid} - d^{k+1}_R) - (s - d^{mid} - d^{k+1}_R)) + 2 \cdot k \cdot N) \\ &= \quad \frac{1}{P^{k-1}} \cdot ((t-s) + 2 \cdot k \cdot N) \\ &= \quad \frac{1}{P^{k-1}} \cdot (t-s) + 2 \cdot P \cdot N \cdot \frac{k}{P^k} \\ C^{R(k)}_{i_k,p_k,j}(t) - C^{R(k)}_{i_k,p_k,j}(s) \quad &= \quad \lfloor \frac{t-s}{P^{k-1}} \rfloor \\ &> \quad \frac{1}{P^{k-1}} \cdot (t-s) - 1\end{aligned}$$

Thus:

$$\begin{aligned}q^{R(k)}_{i_k,p_k,j}(t) \quad &= \quad \max_{0 \leq s \leq t}[(A^{R(k)}_{i_k,p_k,j}(t) - A^{R(k)}_{i_k,p_k,j}(s)) - (C^{R(k)}_{i_k,p_k,j}(t) - C^{R(k)}_{i_k,p_k,j}(s))] \\ &< \quad \max_{0 \leq s \leq t}[(\frac{1}{P^{k-1}} \cdot (t-s) + 2 \cdot P \cdot N \cdot \frac{k}{P^k}) - (\frac{1}{P^{k-1}} \cdot (t-s) - 1)] \\ &= \quad \max_{0 \leq s \leq t}[2 \cdot P \cdot N \cdot \frac{k}{P^k} + 1] \\ &= \quad 2 \cdot P \cdot N \cdot \frac{k}{P^k} + 1\end{aligned}$$

We set $d_B = 2 \cdot k \cdot N + P^k$, then it holds:

$$\begin{aligned}&B^{R(k)}_{i_k,p_k,j}(t + d_B) - A^{R(k)}_{i_k,p_k,j}(t) = \\ &\qquad = \quad \min_{0 \leq s \leq t + d_B}[(A^{R(k)}_{i_k,p_k,j}(s) + (C^{R(k)}_{i_k,p_k,j}(t + d_B) - C^{R(k)}_{i_k,p_k,j}(s))] - A^{R(k)}_{i_k,p_k,j}(t)\end{aligned}$$

$$
\begin{aligned}
&= \min_{0\le s\le t+d_B}[(C^{R(k)}_{i_k,p_k,j}(t+d_B)-C^{R(k)}_{i_k,p_k,j}(s))-(A^{R(k)}_{i_k,p_k,j}(t)-A^{R(k)}_{i_k,p_k,j}(s))]\\
&= \min\left[\begin{array}{rl} \min_{0\le s\le t} & [(C^{R(k)}_{i_k,p_k,j}(t+d)-C^{R(k)}_{i_k,p_k,j}(s))-(A^{R(k)}_{i_k,p_k,j}(t)-A^{R(k)}_{i_k,p_k,j}(s))],\\ \min_{t<s\le t+d_B} & [(C^{R(k)}_{i_k,p_k,j}(t+d)-C^{R(k)}_{i_k,p_k,j}(s))-(A^{R(k)}_{i_k,p_k,j}(t)-A^{R(k)}_{i_k,p_k,j}(s))]\end{array}\right]
\end{aligned}
$$

For $(0 \le s \le t)$, it holds:

$$
\begin{aligned}
C^{R(k)}_{i_k,p_k,j}(t+d_B)-C^{R(k)}_{i_k,p_k,j}(s) &> \frac{1}{P^{k-1}}\cdot(t+d_B-s)-1\\
&= \frac{1}{P^{k-1}}\cdot(t-s)+\frac{P}{P^k}\cdot d_B-1\\
A^{R(k)}_{i_k,p_k,j}(t)-A^{R(k)}_{i_k,p_k,j}(s) &\le \frac{1}{P^{k-1}}\cdot(t-s)+2\cdot P\cdot N\cdot\frac{k}{P^k}
\end{aligned}
$$

For $(t < s \le t+d_B)$, it holds:

$$
\begin{aligned}
C^{R(k)}_{i_k,p_k,j}(t+d_B)-C^{R(k)}_{i_k,p_k,j}(s) &\ge 0\\
A^{R(k)}_{i_k,p_k,j}(t)-A^{R(k)}_{i_k,p_k,j}(s) &\le 0
\end{aligned}
$$

Thus:

$$
\begin{aligned}
&B^{R(k)}_{i_k,p_k,j}(t+d_B)-A^{R(k)}_{i_k,p_k,j}(t)\ge\\
&\ge \min[\min_{0\le s\le t}[(\frac{1}{P^{k-1}}\cdot(t-s)+\frac{P}{P^k}\cdot d_B-1)-(\frac{1}{P^{k-1}}\cdot(t-s)+2\cdot P\cdot N\cdot\frac{k}{P^k})],0+0]\\
&\ge \min[\min_{0\le s\le t}[\frac{P}{P^k}\cdot d_B-1-2\cdot P\cdot N\cdot\frac{k}{P^k}],0]\\
&\ge \min[\min_{0\le s\le t}[\frac{P}{P^k}\cdot(d_B-2\cdot k\cdot N)-1],0]\\
&\ge \min[\min_{0\le s\le t}[\frac{P}{P^k}\cdot P^k-1],0]\\
&\ge \min[\min_{0\le s\le t}[P-1],0]\\
&\ge 0
\end{aligned}
$$

That is:

$$
d^{R(k)} \le d_B = 2\cdot k\cdot N+P^k
$$

The above equation also implies that d^k_r is finite and the set $B_j(t')$ passes through stage $k-1$ of the routing Banyan with a maximum delay of $d^{R(k)}$ cell times, i.e.

$$
\begin{aligned}
\bigcup_{i_k,p_k} B^{R(k)}_{i_k,p_k,j}(t'+d^{mid}+d^{k+1}_R+d^{R(k)}) &\supseteq B_j(t')\Rightarrow\\
\bigcup_{i_k,p_k} B^{R(k)}_{i_k,p_k,j}(t'+d^{mid}+d^{k}_R) &\supseteq B_j(t')
\end{aligned}
$$

Appendix B

Simulation Issues

B.1 Bursty Traffic Model

The bursty traffic model used in this paper is a small variant of the model described in [33, section C.4.1]. It is based on the two-state Markov chain shown in fig. B.1.

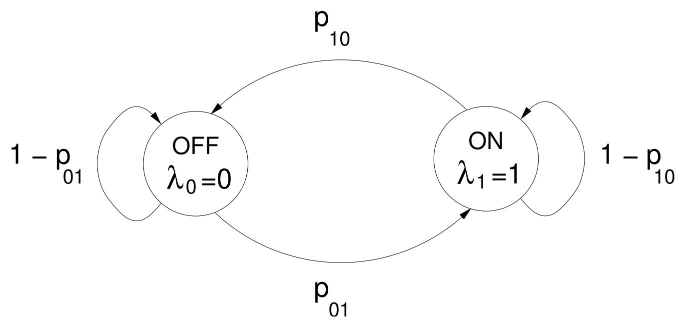

Figure B.1: *Two-state Markov chain for generating bursty traffic.*

In the variant used in this paper *(a)* the ON state has a minimum length of one burst, *(b)* the sojurn in the OFF state can be of zero length, *(c)* each burst holds for one sojurn in the ON state. Let λ denote the average traffic load, B denote the average burst size, and $\overline{T_{on}}$ the average ON state length. We choose to allow for sojurns in th OFF state of zero length, because if the OFF state has a minimum length of one burst, which in turn has a minumum length of one packet slot, then the highest achieved load is:

$$\lambda = \frac{B}{B+1}$$

We choose to identify a burst with a sojurn in the ON state, because in that case it holds:

$$B = \overline{T_{on}}$$

In our case, for the average length of the ON and OFF states, it holds:

$$\overline{T_{on}} = \frac{1}{p_{10}}$$

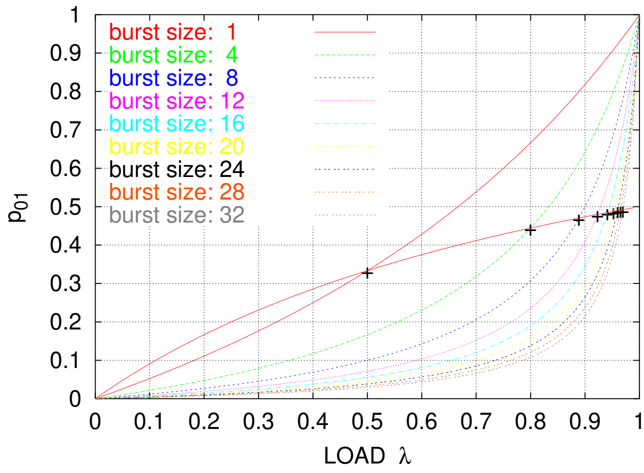

Figure B.2: *Value of p_{01} as a function of λ and B. The figure also highlights the value of p_{01} for the maximum achievable load when the OFF state has a minimum length of one burst.*

$$\overline{T_{off}} = \frac{1 - p_{01}}{p_{01}}$$

For the state probabilities, it holds:

$$\begin{aligned} P_0 &= \frac{\overline{T_{on}}}{\overline{\overline{T_{on}} + \overline{T_{off}}}} \\ P_1 &= \frac{\overline{T_{off}}}{\overline{\overline{T_{on}} + \overline{T_{off}}}} \end{aligned}$$

That is:

$$\begin{aligned} P_0 &= \frac{p_{10} - p_{01} \cdot p_{10}}{p_{01} + p_{10} - p_{01} \cdot p_{10}} \\ P_1 &= \frac{p_{01}}{p_{01} + p_{10} - p_{01} \cdot p_{10}} \end{aligned}$$

And the average traffic load equals:

$$\lambda = P_1$$

Thus, given parameters B and λ, we compute:

$$\begin{aligned} p_{10} &= \frac{1}{B} \\ p_{01} &= \frac{\lambda}{(1 - \lambda) \cdot B + 1} \end{aligned}$$