FORTH-ICS / TR-303 March 2002

# Benes Fabrics with Internal Backpressure: First Work-in-Progress Report

*Georgios Sapountzis*[1,2] *and Manolis Katevenis*[1,2]

**Abstract**

Buffered switching fabrics are a method for building packet switches with a very large number of ports. Multilane backpressure in the switching fabric allows the switching elements to only use on-chip buffer memory, while the majority of the packets are buffered at the inputs, in virtual-output queues (VOQ). In this report we extend the above ideas to the Benes network which is the lowest-cost topology that is known to yield operation free of internal blocking. We study simple, hardware-efficient cell distribution and resequencing algorithms. We present our work towards quantifying the internal delays caused by cell distribution – delays that do not result in throughput limitations, anyway. We also identify the points in the fabric where input and output contention are resolved. This is a work-in-progress report: our work is not yet completed or finished, and we identify several open questions that remain; in some cases we purposefully use simpler models than the actual packet system. Finally we provide preliminary results from packet simulations to complement the analysis.

**Keywords:** Benes fabric, internal backpressure, round-robin cell distribution, distributed scheduling

[1]ICS-FORTH, P.O. Box 1385, GR 711 10 Heraklion, Crete, Greece. E-mail: {sapunjis,katevenis}@ics.forth.gr
[2]Department of Computer Science, University of Crete, Heraklion, Crete, Greece.

# Benes Fabrics with Internal Backpressure: First Work-in-Progress Report

Georgios Sapountzis, Manolis Katevenis

{sapunjis, katevenis}@ics.forth.gr

Institute of Computer Science (ICS)

Foundation for Research and Technology - Hellas (FORTH)

P.O. Box 1385, Heraklion, Crete, GR 711 10, Greece

http://archvlsi.ics.forth.gr/bpbenes/



Mar 2002

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Buffered switching fabrics are a method for building packet switches with a very large number of ports. Internal backpressure in buffered switching fabrics has been shown to provide high performance at low cost [6]. In particular, (multilane) backpressure in the switching fabric allows the switching elements to only use on-chip buffer memory, while the majority of the packets are buffered at the inputs, in virtual-output queues (VOQ). This research concerns the extension of the above ideas to the Benes network [1], which is the lowest-cost topology that is known to yield operation free of internal blocking.

The overall purpose of this work is to use the class of non-blocking multi-stage fabrics with $2 \cdot \log N$ stages as a basis for building a large composite packet switch using small packet switches as building blocks. Thus, the objectives for such an architecture are more demanding than what they would be if we just wanted the resulting fabric to be non-blocking. Specifically:

- The $N \times N$ fabric should be non-blocking: full input/output link utilization for any traffic pattern that satisfies the input and output link constraints.
- The $N \times N$ fabric should provide QoS guarantees: ideally, in the presense of input/output contention the fabric should serve the flows according to their weighted max-min fair share.
- Low delay: minimize the time required for a flow to get access to its reserved bandwidth.
- The implementation cost of each switching element should be low: the operations that each switching element must perform and the state that it must maintain should be minimal; in particular, the state should scale not worse than linearly with the number of inputs (outputs) of the $N \times N$ fabric.

We study simple, hardware-efficient cell distribution and resequencing algorithms and their interaction with backpressure: in order for the Benes network to be non-blocking, traffic must be carefully distributed to the multiple internal paths leading to the desired output, and later resequenced. We also perform a study in order to identify the points in the architecture where contention is resolved, and make it possible for recently developed distributed scheduling algorithms to be employed over the proposed architecture. The rest of this report presents the cell distribution method and the analysis of contention resolution; we also present methods to reduce the complexity of the architecture.

This is a Work-in-Progress Report, there are cases where we simplify the analysis in one or more of the following dimensions: (i) traffic model (fluid vs. packet), (ii) feedback delay (zero vs. non-zero, finite), (iii) input conditions (stationary vs. changing), (iv) buffer size (infinite vs. finite). Input conditions refer to the arriving rates of flows, moreover, even with infinite buffers we may employ feedback, not for preventing buffer overflow, but for rate matching. We provide packet simulations to complement the analysis. *The report contains several open questions that we work on, and several hypotheses based on intuition, which, however, we have not yet verified or checked by packet simulations. More reports on our work will follow.*

In a real system, each switching element of the fabric will be $P \times P$ (e.g. $P = 32$), and ports may be bidirectional. In this report, to simplify the discussion, most of the time we use $2 \times 2$ switches and assume cell flow in a single direction. We also assume fixed-size packets (cells), which is a common technique employed in high-speed switches, and a single flow from each input to each output of the fabric (microflow). However, we take care so that the latter assumption can be loosened and the architecture can be extended to multiple virtual lanes per input-output pair (e.g. multiple priority levels).

# Chapter 2

# Generic Architecture

## 2.1 Traffic Distribution

Assuming a fluid traffic model, it is possible to build an internally non-blocking network with $2 \cdot \log N$ stages. Such a network can be constructed recursively and the resulting topology is known as the Benes network. The $N \times N$ Benes network consists of two $\frac{N}{2} \times \frac{N}{2}$ Benes subnetworks, $\frac{N}{2}$ switches connected to the inputs of the two subnetworks (called the input switches), and $\frac{N}{2}$ switches connected to the outputs of the two subnetworks (called the output switches).

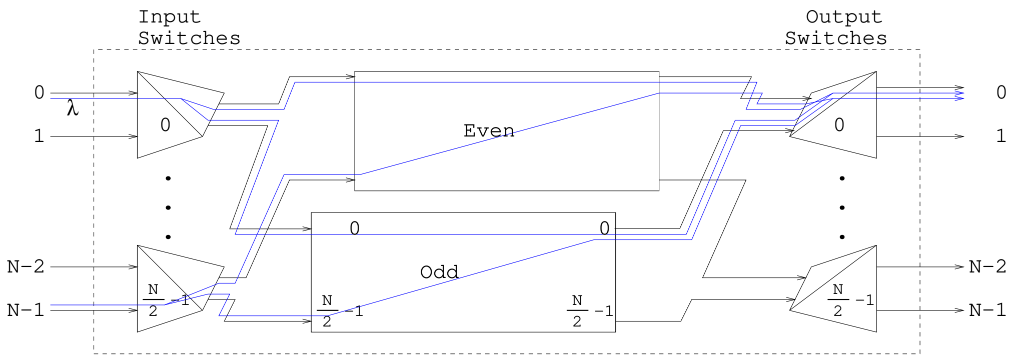



Figure 2.1: *Recursive construction of an $N \times N$ Benes network.*

Let $\lambda_{i,j}$ to 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 $2 \times 2$ input switch connected to input $i$ must equally distribute $\lambda_{i,j}$ among its two outputs. The output switch connected to output $j$ receives $\frac{1}{2} \cdot \lambda_{i,j}$ on each of its inputs, reconstructs $\lambda_{i,j}$ and routes it to the appropriate output. The above discipline must be followed by the input and output swithes of each Benes subnetwork for the portion of traffic that reaches the inputs of that subnetwork and for the flows as defined by the input-output pairs of that subnetwork. Thus, traffic $\lambda_{i,j}$ goes through $\log N$ stages of distribution and $\log N$ corresponding stages of reconstruction.

Unrolling the recursion in Fig. 2.1, for N = 8, results in the topology shown in Fig. 2.2. 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 *routing* networks [10], since the first Banyan distributes incoming cells over the $N$ links at the heart of the network (the set of the $N$ links can also be seen as a virtual wide link of throughput $N$), and the second Banyan routes each cell that arrives through the wide link to the appropriate output link of throughput 1 cell per cell time.

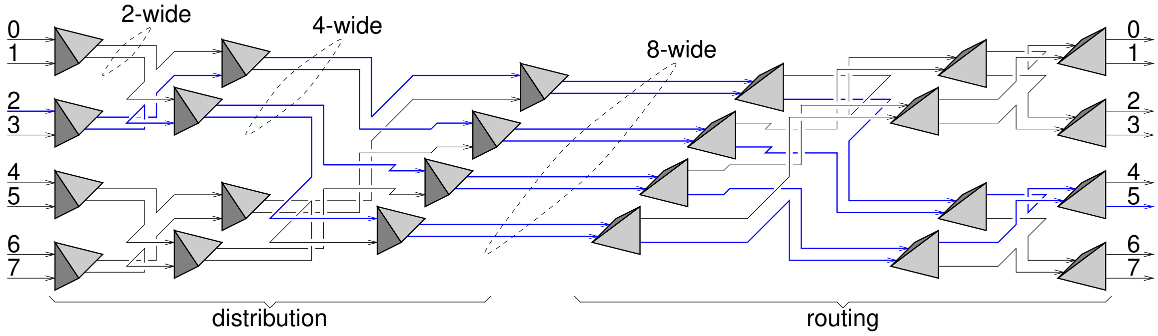



Figure 2.2: *Benes network with size* $8 \times 8$*. The figure also shows distribution and reconstruction of traffic* $\lambda_{2,5}$ *in order for the network to be non-blocking, assuming* $\lambda_{2,5}$ *is the only active flow in the network.*

The straight-forward application of the method that ensures non-blocking operation on a packet network is *bit slicing*: each distribution switching element equally divides the bits of every incoming cell among its two outputs. The above scheme equally distributes incoming traffic among the two Benes subnetworks but suffers from non-scalability: the links at the heart of the network carry traffic in units of $\frac{L}{N}$, where $L$ is the original cell size; as a result, switches at the heart of the network operate $N$ times faster.

A number of methods have been proposed in order to approximate "even" traffic distribution in a cell system and still operate at the cell level: randomized [11], adaptive [3], delay equalization [12]. In section 3, we present the proposed method for cell distribution and compare it with the aforementioned methods.

## 2.2 Flow Groups

In order to approximate even traffic distribution, distribution switching elements distribute microflows to their two outputs. Thus, the number of flows doubles in each stage of the distribution Banyan. For an $N \times N$ fabric, there are $2^{\log N} \cdot N = N^2$ microflows passing through each of the middle links. If our scheme needs per-(micro)flow state (which it will likely need), then, complexity of $O(N^2)$ will result, which is prohibitive for large $N$.

Flow *merging* (or *aggregation*) is a method for reducing the number of flows. In order to ensure correctness and avoid performace degradation: (1) The flow merging method should be compatible with the distribution and resequencing of traffic. Specifically, traffic is distributed and resequenced on a per-flow (merged flow) or per-microflow (input-output pair) basis. We may have to split merged flows in order to expose original flow information to traffic resequencing. (2) The flow merging method should be compatible with the flow control algorithm. For example, hop-by-hop credit-based flow control combined with merging flows destined to different outputs suffers from *HOL blocking*.

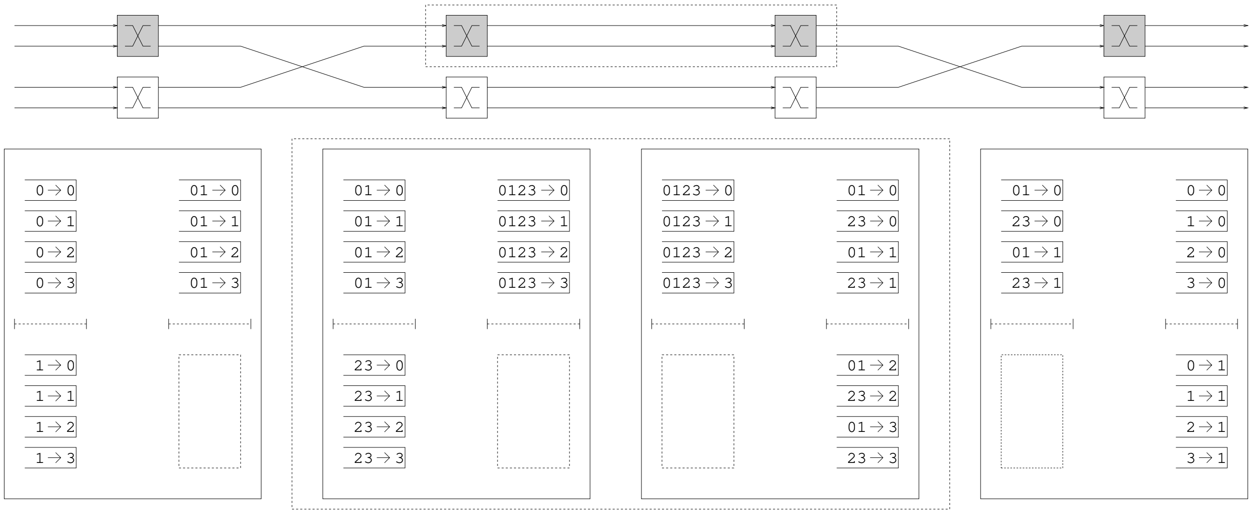



Figure 2.3: *A* $4 \times 4$ *fabric and the flow groups at the inputs and outputs of the shaded switching elements for the per-output flow merging case.*

Fig. 2.3 and 2.4 show the two alternatives for flow merging and splitting: (a) *per-output* flow merging and (b) *hierarchical* flow merging. It is important to note that the figures do not show the *cell queues* that the switching elements must maintain, but the *flow groups* that the flow control protocol must operate on when it sends/receives cells and credits. The rest of this section explains the two methods in more detail.

Per-output merging merges flows that are destined to the same *final* destination, that is for the same output of the $N \times N$ fabric. For example, as shown in Fig. 2.3, per-output merging merges flow $0 \rightarrow 0$ with $1 \rightarrow 0$ into $01 \rightarrow 0$. Each distribution switching element (in the first half of the fabric) performs flow merging and cell distribution. Flow splitting, in the second half of the fabric, is necessary beacuse cell distribution is performed per-output at each input, thus, cell resequencing must be performed per-input at each output. With per-output flow merging, the merging factor is $P = 2$, and the number of flows through each link is reduced down to $N$. Moreover, hop-by-hop credit-based flow control and buffers of size 1 may still allow full utilization of output links.

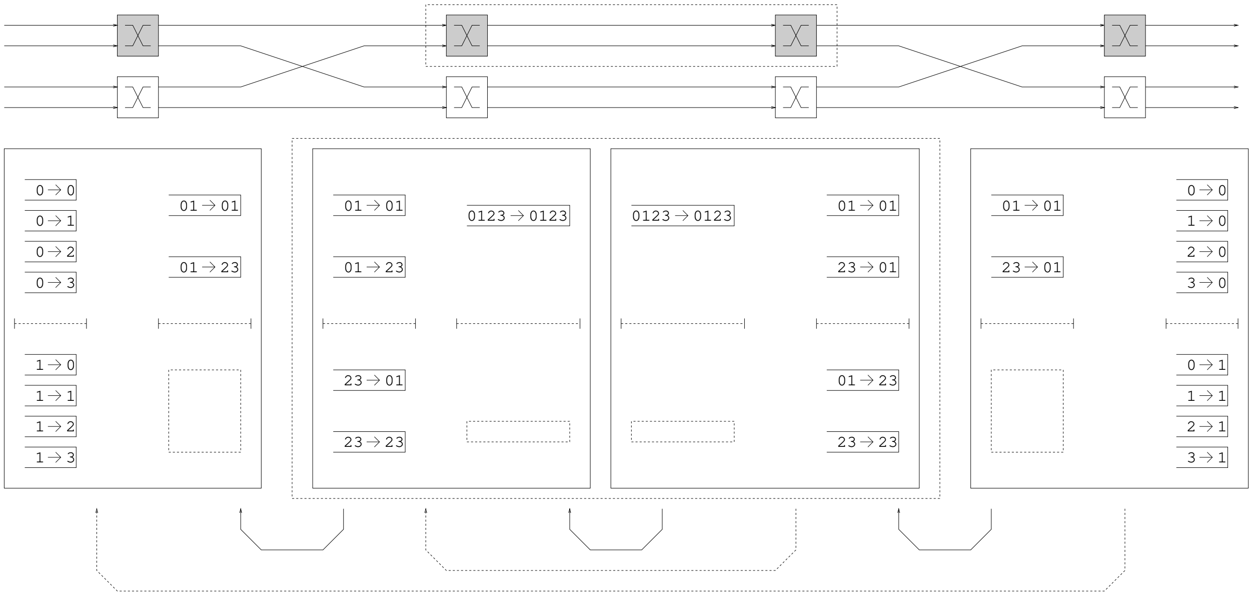



Figure 2.4: *A $4 \times 4$ fabric and the flow groups at the inputs and outputs of the shaded switching elements for the hierarchical flow merging case.*

We now turn to hierarchical flow merging. Hierarchical flow merging, follows the recursive construction of the Benes network (see Fig. 2.1). Specifically, the input (output) switching elements of a $K \times K$ Benes subnetwork maintain state for the $K$ flows destined to each output (arriving from each input) of the $K \times K$ subnetwork only. For example, as shown in Fig. 2.4, hierarchical merging merges all flows $0 \to 0,\ \ 1 \to 0,\ \ 0 \to 1,\ \ 1 \to 1$ into one flow group $01 \to 01$. The merging factor is $P^2 = 4$, and the number of flows through each link at stage $s = \log \frac{N}{K}$ is reduced down to $K = \frac{N}{2^s}$. Hierarchical merging merges flows that are destined to different *final* destinations, thus - in order to eliminate HOL blocking - we need to use "hierarchical backpressure" [5, section III-D]: one level of backpressure is used from each splitting point to the corresponding merging point, which results in a total of $\log N$ levels of backpressure. Moreover, since the flow control feedback delay is $2 \cdot \log N$ for the last stage of the routing Banyan, buffers of size $2 \cdot \log N$ may be required for full link utilization.

| | Levels of backpressure | Number of flow groups | Per-stage, per-flow buffer size |
|---|---|---|---|
| Per-output | 1 | $N$ | 1 |
| Hierarchical | $\log N$ | $\frac{N}{2^s}$ | up to $2 \cdot \log N$ |

Table 2.1: *Summary of comparison of per-output and hierarchical flow merging.*

Table 2.2 summarizes the initial comparison of the flow merging methods. Per-output flow merging seems much simpler to implement. Hierarchical flow merging needs to be further analysed with regard to implementation cost (levels of backpressure, total buffer space) and performance. In this report we focus on per-output flow merging and present organizations based on hierarchical flow merging when they offer some insight in the operation of the system.

## 2.3 Logical Buffer Organization

In this section we present the logical buffer organization of the distribution and routing switching elements. The organization presented in this section will be later simplified as we take advantage of the properties guaranteed by the protocols implemented by the switching elements. Fig. 2.5 shows a generic logical buffer organization that is compatible with the flow groups shown in Fig. 2.3.

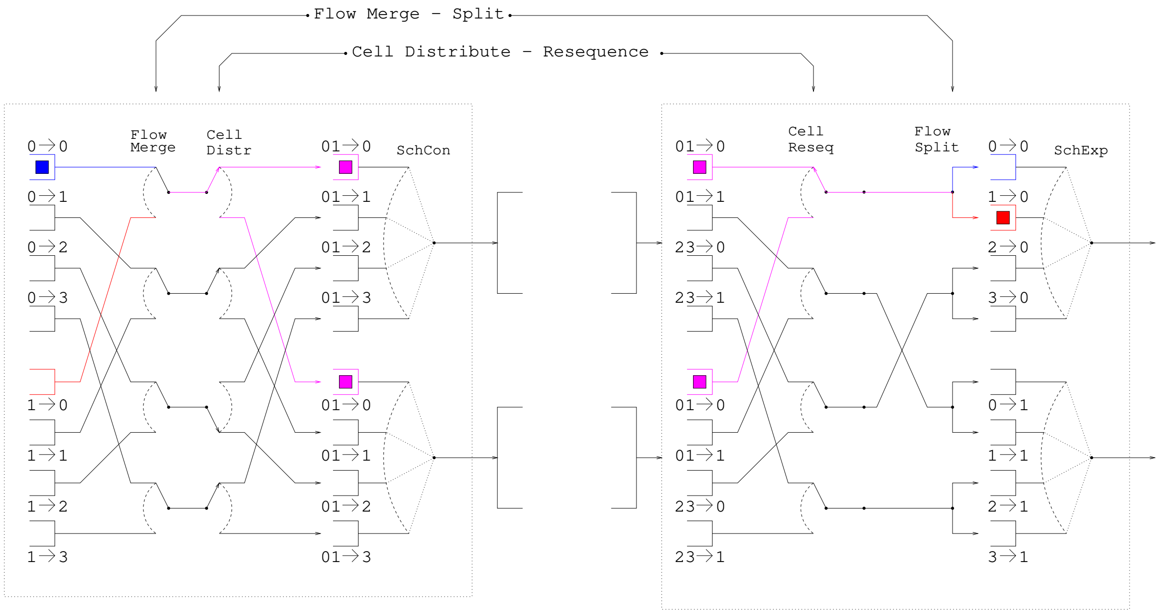



Figure 2.5: *Logical buffer organization of a distribution and the corresponding routing switching element for per-output flow merging. The figure also shows the merging and splitting of flows* $0 \rightarrow 0$ *and* $1 \rightarrow 0$*, and the distribution and resequencing of the cells of the merged flow. The various active components are also shown:* `Merge`*,* `Distr`*,* `SchCon`*,* `Reseq`*,* `Split`*, and* `SchExp`*.*

A distribution switching element must perform flow merging and cell distribution. These two tasks can be performed in any order. However, flow merging before cell distribution allows for a better approximation to even traffic distribution. This is the reason for the choice made in Fig. 2.5.

The active components are meant to show the functionality implemented by the switching elements. The queues are meant to be logical queues (do not contain cell data, but pointers to the actual buffer space where cells are queued) and, in later sections, we present simpler logical buffer organizations where some of the logical queues have been eliminated.

## 2.4 Queueing Points

In this section we give an overview of the queueing points in an $N \times N$ fabric. We identify two reasons that cause cells to delay within the fabric: (i) input/output contention, and (ii) traffic quantization. We use Fig. 2.6 for identifying the queueing points and explaining the cause of queueing.

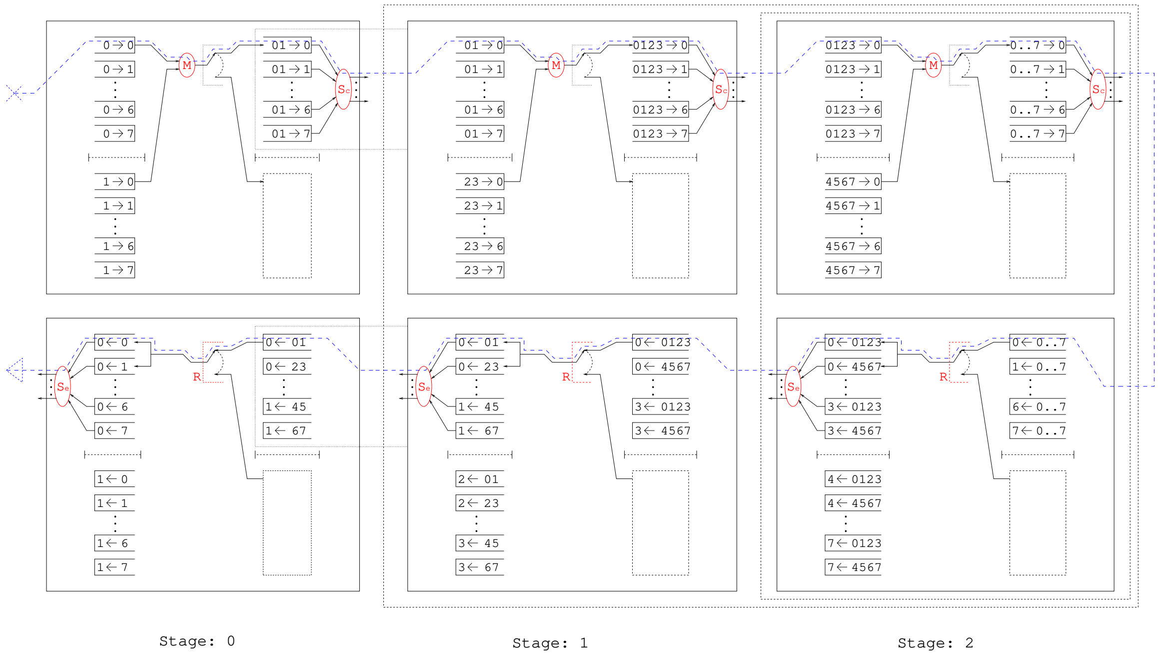



Figure 2.6: *Possible contention points for a* $8 \times 8$ *fabric with per-output flow merging. The figure only shows the first row of the switches of the network. The upper row shows the distribution switches, while the lower row shows the routing switches.* `M` *stands for* `Merge`, $\mathtt{S_c}$ *stands for* `SchCon`, `R` *stands for* `Reseq`, *and* $\mathtt{S_e}$ *stands for* `SchExp`.

Traffic quantization causes delays that would not be present under the fluid traffic model. The characteristics of this form of delays depend on the cell distribution method. Consider an $8 \times 8$ 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 be delayed at: (i) the `SchCon` queues of the distribution switching elements of stage 0, (ii) the `SchExp` queues of the routing switching elements of stage 1, and (iii) the `Reseq` queues of the routing switching elements of stage 0.

We now provide some examples that demonstrate the above cases. Suppose that two cells that belong to flows $0 \to 0$ and $1 \to 7$ arrive at the fabric at the same cell time, and the cell distribution method decides that they need both be forwarded through the even Benes subnetowrk, then case (i) arises. Now, suppose that two cells that belong to flows $0 \to 0$ and $7 \to 1$ arrive at the fabric at the same cell time, and the cell distribution method decides that are both forwarded through the even 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 (ii) arises.

With regard to case (iii), suppose that two cells $c_i$ and $c_{i+1}$ of flow $0 \to 0$ 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 (iii) 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 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, 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 would be performed at the queues of `SchExp` but the fabric would not be able to match the service of flows at the inputs and outputs and stabilize at the fabric-wide max-min fair share for every flow. With finite buffers and feedback, the output constraints would cause packets to wait at the `SchExp` queues and because of the feedback, at `SchCon` and `Merge`. The roles of `Merge`, `SchCon` and `SchExp` in the scheduling architecture and their interactions thereof are studied in chapter 4.

In the following sections we study the Benes fabric with regard to

- the cell distribution method and the conditions under which a feasible traffic pattern faces internal delays – we show that this is due to the distributed nature of the cell distribution method and does not result in throughput limitations.
- the points where contention is resolved – we study how the active components of the distribution and routing switching elements can coordinate to allocate the capacity of the fabric to the $N \times N$ competing flows.
- the logical buffer organization – we simplify the logical buffer organization of the switching elements to simpler configurations maintaining their equivalence to the initial configurations with regard to the flow control protocol and the scheduling architecture.

# Chapter 3

# Cell Distribution and Resequencing

## 3.1 Introduction

In a Benes network 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 source and destination, and information maintained locally, determines which Benes subnetwork to forward the cell through.

In the rest of this chapter, we present the cell distribution method and an initial performance analysis, and then, we prove that cell distribution is actually deadlock-free.

## 3.2 Round-Robin Deterministic Cell Distribution

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 [7] is to forward the cells of each microflow $i \to j$ *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 packets of microflow $i \to j$ 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 microflow from the same subnetwork through which the distribution switching elements started forwarding the cells of that microflow. 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 [4] in order to eliminate the internal speedup requirement.

Consider microflow $i \to j$, 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 $i \to j$ *currently* on some path of the fabric starting (at the distribution switching element) just *after* the cell distribution point towards the even Benes subnetwork and terminating (at the routing switching element) just *before* the cell resequencing point from the even Benes subnetwork. 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 set of paths are disjoint.

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

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

**Proof 1** *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 $C^e_{ij}$ 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. Similarly, we can prove the equation for the case where any of the sets $C^e_{ij}$ and $C^o_{ij}$ is empty.*

Eq. 3.1 implies:

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

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

Eq. 3.1 means that round-robin cell distribution perfectly equalizes the per-microflow traffic volume among the two Benes subnetworks. However, eqs. 3.2 and 3.3 show that round-robin cell distribution does not perfectly equalize the traffic volume coming from each input or destined to each output of the Benes subnetworks. Inefficiencies of the cell distribution method at stage $s$ result in backlog at (i) the `SchCon` components of the distribution switching elements of stage $s$, called *input* backlog in this report, (ii) the `SchExp` components of the routing switching elements of stage $s+1$, called *output* backlog in this report.

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. However, as noted in [8], output backlog 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 backlog, are summarized below:

- `output` backlog: 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 of the subnetwork.
- `input` backlog: 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.

### 3.2.1 Input/Output Backlog

In this section we study input and output backlog. We first consider input backlog, the result of this section is identical to the result of [4, Lemma 3]. However, we repeat it here in the context of the Benes fabric in order for the report to be self-contained and because it was developed independently.

**Theorem 2** *Under the following assumptions: (i) infinite buffers, (ii) no feedback, (iii) round-robin cell distribution, (iv) work-conserving* `SchCon`*'s, the maximum backlog at any* `SchCon` *is* $(1 - \frac{1}{P}) \cdot N$.

**Proof 2** *Consider a* $P \times P$ *distribution switching element and a time window* $[0, t]$ *such that the* $p^{th}$ `SchCon` *is* continuously *backlogged in the window* $[0, t]$. *Let* $A^p(t)$ *denote the total number of cells that the cell distribution method has decided that they should be forwarded through the* $p^{th}$ `SchCon` *in the window* $[0, t]$, *and* $B^p(t)$ *the backlog of the* $p^{th}$ `SchCon` *at time* $t$. *Also, let time* $0$ *be the beginning of the backlog period, i.e.* $B^p(0^-) = 0$.

*Consider the cells of a given flow that have arrived at the* $p^{th}$ `SchCon`. *Because of round-robin cell distribution, for each cell of the flow, except from the last cell of the flow, there is a "sibling" cell that has arrived at each of the other* $(P-1)$ `SchCon`*'s within the time window* $[0, t]$. *Let* $S^p(t)$ *be the number of cells that have arrived at the* $p^{th}$ `SchCon` *in the window* $[0, t]$ *and have* $(P-1)$ *"sibling" cells, and* $L^p(t)$ *be the number of cells that have arrived at the* $p^{th}$ `SchCon` *in the window* $[0, t]$ *and do not have* $(P-1)$ *"sibling" cells.*

*Since the total number of cells that have arrived at the switching element within the window* $[0, t]$ *is at most* $P \cdot t$, *it holds:*

$$P \cdot S^p(t) + 1 \cdot L^p(t) \leq P \cdot t \Rightarrow S^p(t) \leq t - \frac{L^p(t)}{P}$$

*Since there are* $N$ *flows through the switching element (after flow merging) it holds:* $L^p(t) \leq N$ *. Thus:*

$$A^p(t) = S^p(t) + L^p(t) \leq (t - \frac{L^p(t)}{P}) + L^p(t) = t + L^p(t) \cdot (1 - \frac{1}{P}) \leq t + N \cdot (1 - \frac{1}{P})$$

*Lastly, since* `SchCon` *is work-conserving and is continuously backlogged in the window* $[0, t]$, *it forwards* $t$ *cells in* $t$ *cell times, thus:*

$$B^p(t) = A^p(t) - t \leq N \cdot (1 - \frac{1}{P})$$

In case the distribution switching elements first performed cell distribution and then flow merging, it would be:

$$B^p(t) \leq (1 - \frac{1}{P}) \cdot P \cdot N$$

For hierarchical flow merging, it holds:

$$B^p(t) \leq (1 - \frac{1}{P}) \cdot \frac{K}{P}$$

In the above 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/output backlog at the middle stages of the fabric is significantly smaller than the backlog for per-output merging.

We now turn to output backlog. We not only consider the delays caused by output backlog, but all of the delays faced by cells within the routing Banyan.

First of all, note that in a fluid model with infinite buffers, no feedback, work-conserving `SchCon`'s, 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.

Also, note that the fluid model we are considering ( infinite buffers, no feedback, work-conserving `SchCon`'s, even per-flow traffic distribution ) when combined with per-output flow merging is unable 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. However, it is possible to isolate flow groups destined to different outputs from each other: it suffices for the `SchExp`'s to implement round-robin (RR) scheduling, which is the simplest discipline that provides fairness among competing flows. In fact, we use RR scheduling between different flow sets, where each set comprises of all the flows destined to the same output. For example, see Fig. 2.6, the RR schedulers at the outputs of the switching elements at stage 1 of the routing Banyan participating in an $8 \times 8$ fabric and connected to final outputs 0 and 1, consider the following flow sets: $\{\ 01 \rightarrow 0,\ 23 \rightarrow 0,\ 45 \rightarrow 0,\ 67 \rightarrow 0\ \}$ and $\{\ 01 \rightarrow 1,\ 23 \rightarrow 1,\ 45 \rightarrow 1,\ 67 \rightarrow 1\ \}$ .

TODO: Adapt the result for the PPS architecture mimicking a FIFO reference switch on the Benes fabric. May require delay equalization [2].

### 3.2.2 Open Issues

Open issues regarding the topics of section 3.2.1:

- Compare with alternative methods for cell distribution.

- Analysis in case of finite buffers and feedback. It may be better in this case to consider the number of "lost chances" of `SchCon` when compared to the bit-slicing model (see sec. 4.1), instead of "maximum backlog". May need to employ adaptive distribution in this case.

- Study the penalty due to synchronization of the cell distribution pointers. Does our scheme tend to de-synchronize the pointers or accentuate the problem ? For example, do pointers get de-synchronized after an output conflict ?

- Try to reduce the proposed method to randomized routing, either static [11], or dynamic by devising an appropriate pseudo-random function e.g. number of cells of a flow in a window of cell times.

Comments regarding potential improvements:

- When for some flow $i \to j$, it holds $||C^e_{ij}| - |C^o_{ij}|| = 0$, then, the cell distribution method is free to forward the next cell of that flow through any of the two subnetworks.

- Decreasing the bound of eq. 3.2 (which is related to input backlog) at the cost of increasing the bound of eq. 3.1 is not a gain for worst-case delay because it may also increase the bound of eq. 3.3 (which is related to output backlog). Consider the sum of input and output backlog.

- Consider adaptive distribution per distribution stage and resequencing per routing stage (using sequence numbers). This method retains advantages of adaptive distribution, while keeping the resequencing cost low. Adaptive routing should choose the end-to-end path with the minimum backlog. How do I measure the backlog along a path ? Theorhetical work that may provide some insight: "Adversarial Queueing Theory", "The Power of Two Random Choices".

## 3.3 Deadlock Freedom

In this section we show that the Benes fabric is actually deadlock-free. The potential deadlock situation for the case where the distribution switching elements first perform flow merging and then cell distribution is shown in Fig. 3.1. In case the distribution switching elements first performed cell distribution and then flow merging the potential deadlock situation would be very similar, the difference being in the mapping of the functionality to the switching elements participating in the dealock situation.

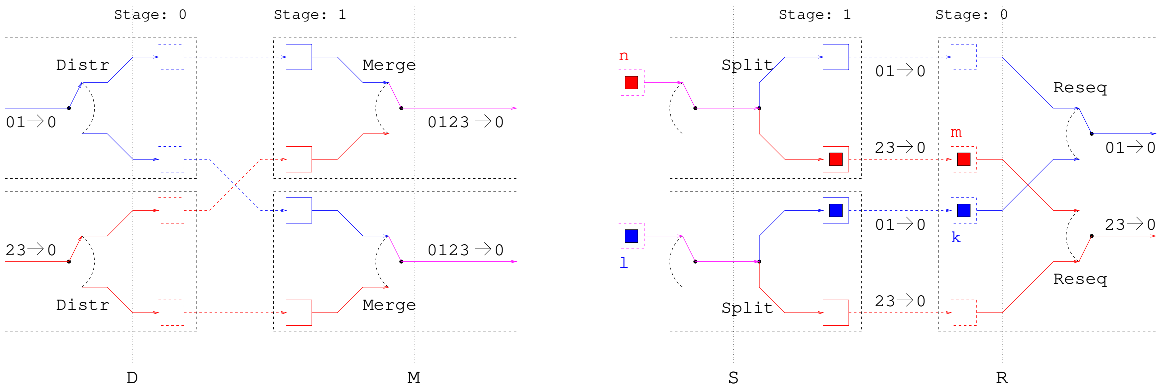



Figure 3.1: *Deadlock situation.*

Let $c_{col}^{seq}$ denote the cell of flow with color "col" and a sequence number of "seq". The deadlock situation arises in the following case. The blue resequencer is waiting for cell $c_{blue}^{k-1}$ and the red resequencer is waiting for cell $c_{red}^{m-1}$. Cell $c_{blue}^{k-1}$ is somewhere in the fabric behind cell $c_{red}^{n}$ and cannot move forward since it needs a buffer hold by cell $c_{red}^{n}$, note that cell $c_{blue}^{k-1}$ could not be on the same path with cell $c_{blue}^{k}$ since they belong to the same flow group and the Benes fabric does not reorder cells within the same flow group; correspondingly for cell $c_{red}^{m-1}$.

**Theorem 3** *If no cells are lost due to electrical noise, all buffers are of size* 1*, and cell forwarding is subject to hop-by-hop credit-based flow control, then any cell distribution method is deadlock-free.*

The proof does not consider the buffers shown in shaded lines; note that, regarding the deadlock situation, considering these buffers is equivalent to considering buffers with size larger than 1.

**Proof 3** *Let $L_{col}^{seq}$ denote the time slot at which cell of color "col" and sequence number "seq" crosses line L. The system operation gurantees a number of ordering relations between various values of $L_{col}^{seq}$.*

***Relation 1*** *Direction: It holds that $(D_c^i < M_c^i < S_c^i)$.*

***Relation 2*** *Steering: It holds that $(D_c^i < D_c^{i+1})$.*

***Relation 3*** *Backpressure: If cells $c_c^i$ and $c_c^j$ with $j > i$ are both forwarded through the same Benes subnetwork, then $(M_c^i < D_c^j)$.*

***Relation 4*** *Recursion: Each of the Benes subnetworks guarantees in order packet delivery, thus, if cells* $c_c^i$ *and* $c_c^j$ *are both forwarded through the same Benes subnetwork, then* $(M_c^i < M_c^j \iff S_c^i < S_c^j)$.

*If we assume that a deadlock arises, then the following ordering relation also holds:*

***Relation 5*** *Deadlock: It holds that* $(S_{red}^n < S_{blue}^{k-1})$, *and* $(S_{blue}^l < S_{red}^{m-1})$.

*The deadlock ordering relation causes a cycle in the time order graph, shown in Fig. 3.2, which is a contradiction. Thus, a deadlock situation cannot arise.*

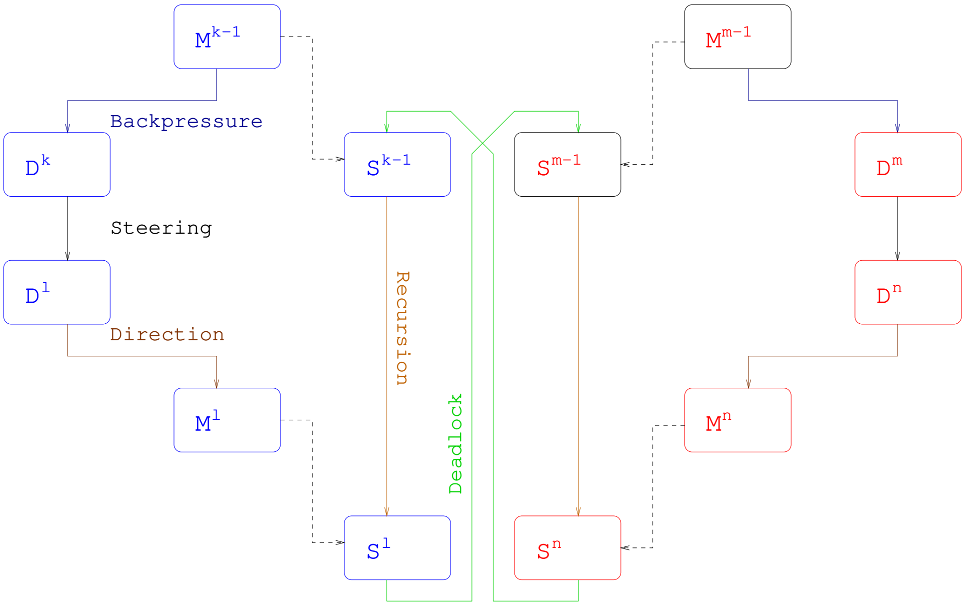



Figure 3.2: *The time order graph. The nodes of the graph represent events of the form* $L_{col}^{seq}$ *and the arcs represent ordering relations between the events. The direction of an arc is from the oldest to the newest event.*

Note that the proof does not assume any special properties for the cell distribution method, the discipline at the flow merging discipline or the discipline after the splitting point. The proof needs to be extended for the case of larger buffer sizes.

# Chapter 4

# 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 first consider an artificial packet system that discards the effects of cell distribution, which we call the "bit-slicing" model, and then consider a system that performs cell distribution, which is the actual "Benes fabric".

## 4.1 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. Note that under this model, if we collapse inputs and outputs of distribution switching elements with identical flow groups, then the distribution switching elements can be considered as having two input links and one output link whose throughput is twice the throughput of each input link. Correspondingly, after collapsing, the routing switching elements can be considered as having one input link and two output links each one having half the throughput of the input link. The bit-slicing model allows us to focus on the contention resolution points of the fabric and develop a scheduling architecture, 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, flow groups as defined with per-output flow merging, and hop-by-hop credit-based flow control. Fig. 4.1 shows a subset of the flow groups traversing an $8 \times 8$ 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

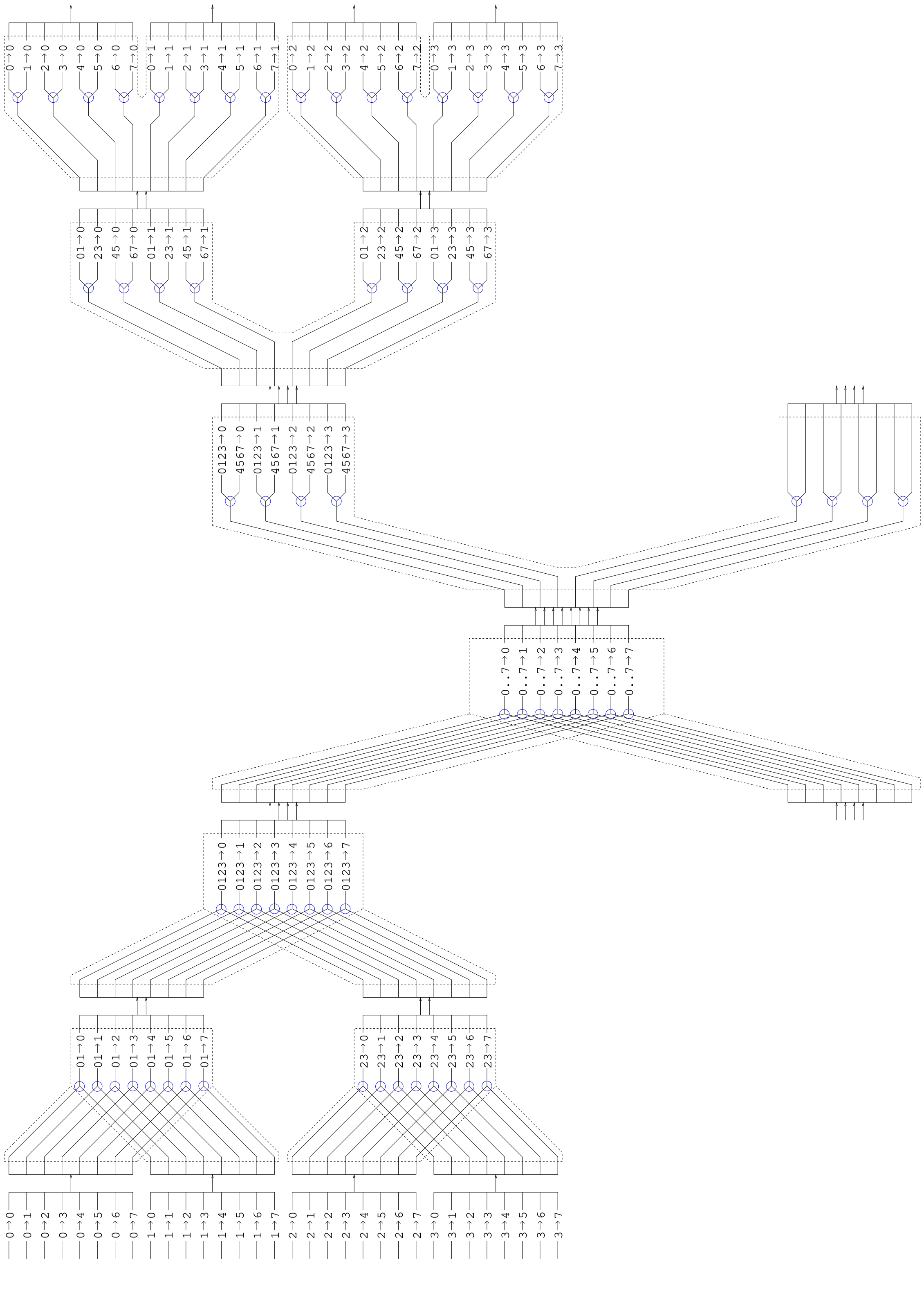


Figure 4.1: *The bit-slicing model. The figure shows a subset of the flow groups for an $8 \times 8$ fabric with per-output flow merging and the distribution and routing switching elements, identified with shaded lines, which these flow groups traverse.*

groups at corresponding 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.

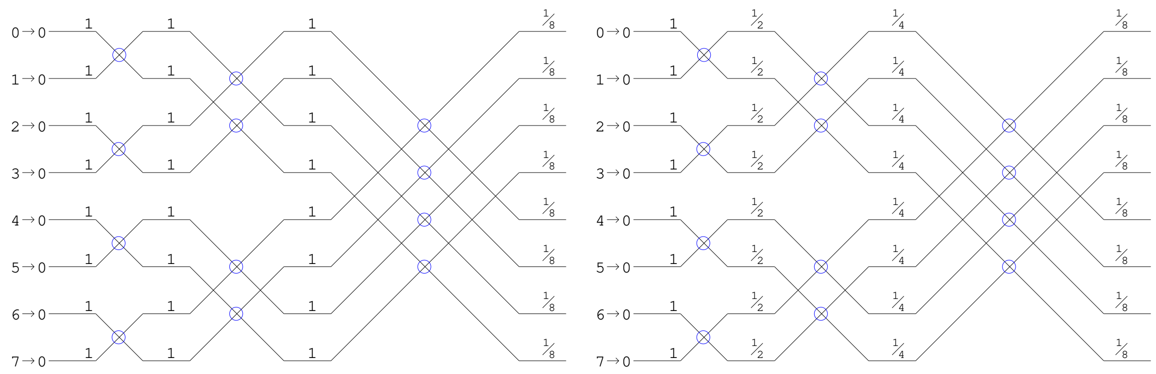
We now present the role of the active components of the switching elements in allocating the capacity of the fabric, and thus delay and bandwidth guarantees, to the $N \times N$ competing flows. In general, allocating the capacity of a switching system with finite buffers reduces to enforcing input and output constraints, and resolving input and output contention. Among the single-stage, single-module switching fabrics, the buffered crossbar architecture [9] 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 $2 \cdot N$ 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.

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, but 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. 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 Banyan and propagated because of 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 at the outputs of the last stage of the distribution Banyan, see Fig. 4.2 (i), or within each stage of the distribution Banyan, see Fig. 4.2 (ii). Note that both cases 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 case shown in Fig. 4.2 (i), the feedback to the heads 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 case shown in Fig. 4.2 (ii) reduces the number of operations per FIFO queue performed in each cell time. However, both cases proactively insert the idle cell times which may turn out to be harmful for the performance of the fabric and may require some form of synchronization between switching elements ports whose flow groups are identical. It remains to be decided whether the idle cell




*(i)* *(ii)*

Figure 4.2: *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 number above the lines denote maximum available rate.*

times cause the performance of the fabric to degrade.

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 `SchExp` within stage 0 of the routing Banyan. However, each time `SchExp` serves a cell from one of the two flows a credit for the 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 `SchExp` 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 `SchExp` would be able to enforce the desired ratios.

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 $0 \to 0$, $7 \to 0$, $0 \to 1$, $6 \to 1$, $1 \to 2$ and $5 \to 2$ 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 of 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 `SchCon` at the head of the queues for flows $01 \to 0$, $01 \to 1$ and $01 \to 2$ is responsible for resolving that form of contention.

In general, input contention appears not only at the input ports of the fabric but at the `SchCon`'s at the outputs of the distribution switching elements, as well. The reason being that the fabric contains a total of $2 \cdot \log N$ buffer stages, and the bandwidth between neigbouring buffer stages is fixed and equal to $N$ cells per cell time. Note that the set of fabric inputs that the flows originate from is common among the flow groups that meet at each `SchCon`.

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. Input contention appears at the input ports and between the stages of the distribution Baynan and is resolved by the `SchCon`'s. There exists a tree of `Merge`'s per fabric output which resolves output contention for that output in a hierarchical fashion.

## 4.2 The Benes Fabric

Consider the effects of cell distribution. The scheduling architecture should be able to provide delay and bandwidth guarantees. In the good scenario, the transient backlog caused by inefficiencies in the cell distribution method will not result in a failure to provide delay guarantees. It will only slightly increase the average delay of flows requiring bandwidth guarantees.

## 4.3 Open Issues

- Distributed PGPS

  Approaches to distributed PGPS are based on composing multiple PGPS schedulers in a hierarchical fashion [9].

  How does cell distribution affect the properties of the composition ?

  Is it possible to achieve fair distribution of excess bandwidth without speedup ?

  TODO: Describe dynamic assignment of weights to flows at `Merge`'s. A problem arises when a flow with high-priority is merged with a flow with low-priority and the state of the high-priority flow changes from inactive to active. The conceptually simpler solution is to have a separate control lane that informs higher levels of the scheduling hierarchy that a high-priority flow has become active.

- Crossbar Scheduling at Concentrator Switches

  By crossbar scheduling (XbarSched) we mean that, at every cell time, the two outputs of a switching element serve cells that originate from different inputs of the switching element. Is it possible to guarantee full link utilization if every switching element uses XbarSched ? How does XbarSched affect cell distribution ? How does XbarSched affect/simplify the flow control protocol ?

- Multi-level Backpressure

  Consider techniques that allow the $\log N$ levels of backpressure to be collapsed into one level. Such a technique will probably come at the cost of increased buffer sizes. Compare the total (flow groups and credits) number of buffers required by single and multi-level backpressure.

# Chapter 5

# Performance Simulation Results

## 5.1 Feasible Traffic

In this section, we present measurements of the performance of the Benes fabric under feasible traffic patterns. We consider two input traffic patterns: (i) independent, identically distributed (i.i.d.) Bernoulli arrivals (i.e. non-bursty), with uniformly-distributed destinations (no overloaded hot-spots), (ii) full permutations. In both cases we measure cell delays with increasing traffic load for various sizes of the fabric, from $8 \times 8$ up to $64 \times 64$, and buffer sizes. In these simple scenarions, the size of the switching elements is $2 \times 2$, all flows have equal weights, the duration of each simulation is 100.000 cell times and the cell delays do not contain the constant term for the delay overhead through the fabric (number of network stages).

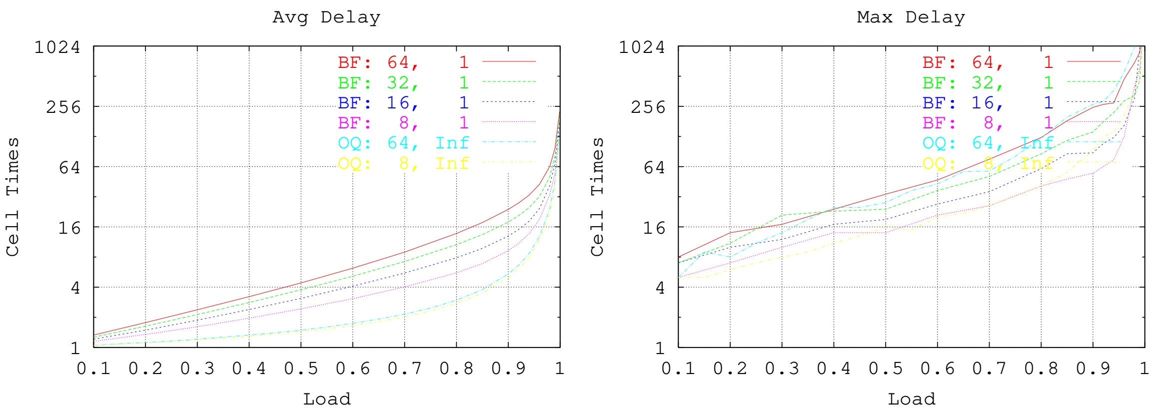



Figure 5.1: *Average and worst-case delay under i.i.d. Bernoulli arrivals.*

The results are shown in Figs. 5.1, and 5.2. "BF" denotes a multi-stage Benes fabric built with $2 \times 2$ switching elements and "OQ" denotes a large, single-stage, output queued switch. The two numbers that

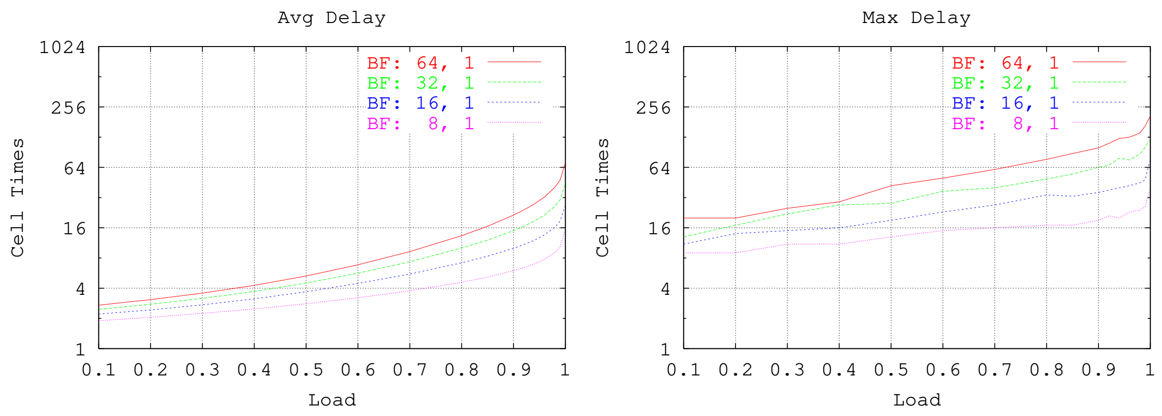



Figure 5.2: *Average and worst-case delay under full permutations.*

follow the name of the system denote the number of inputs/outputs and the buffer sizes – for the fabric case, this is the maximum size of every single queue contained in a switching element.

A few observations about the simulation results follow. Under i.i.d. Bernoulli traffic, the average delay of BF is about 2× the average delay of OQ in the best case ($N = 8$), but increases up-to 4× for larger fabrics ($N = 64$). In the random simulation scenario shown, worst-case delay of BF follows the worst-case delay of OQ. Standard deviation of the delay of BF is about 1 cell time larger than the standard deviation of the delay of OQ. Larger buffers yield only slightly better results, and thus were not included.

Under full permutation traffic, all delays of BF are bounded. Average delay is less than $2 \times N$ cell times. Worst-case delay is approximately $4 \times N$ cell times. Deviation of delay is approximately $\frac{1}{4} \times N$ cell times.

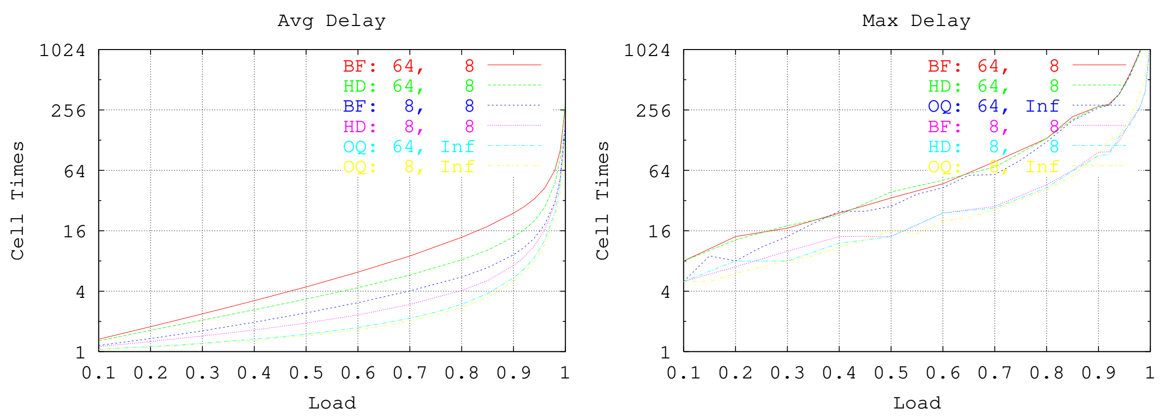



Figure 5.3: *Comparison of per-output and hierarchical cell distribution under i.i.d. Bernoulli arrivals.*

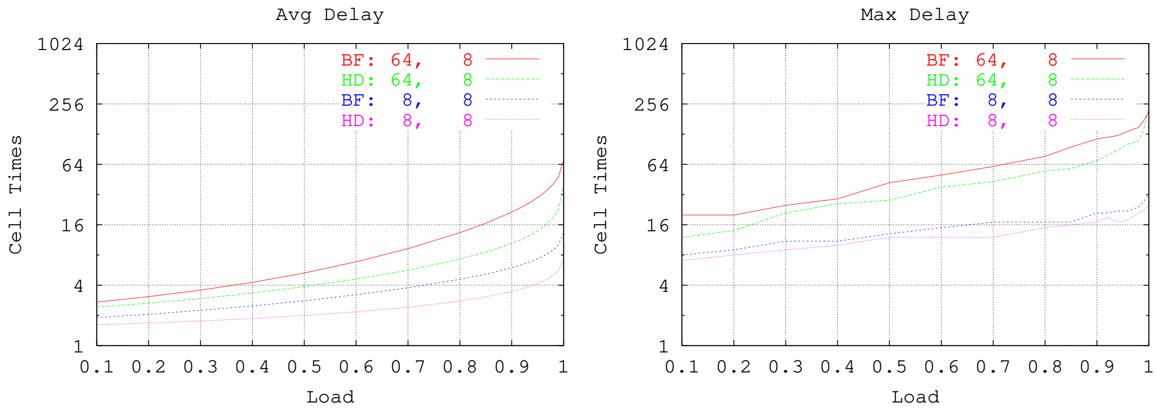



Figure 5.4: *Comparison of per-output and hierarchical cell distribution under full permutations.*

Fig. 5.3 and Fig. 5.4 provide a comparison of per-output (denoted by "BF") and hierarchical cell distribution (denoted by "HD"). By hierarchical cell distribution we mean that we use per-output flow merging but when it comes to cell distribution, flows are treated as if hierarchical flow merging was deployed. The average delays with hierarchical cell distribution, are approximately 1.3× smaller than with per-output cell distribution for a fabric of size $N = 8$. For $N = 64$ the delay reduction fraction is approximately 1.5×. With regard to worst-case delay, we observe that both "BF" and "HD" closely track the worst-case delay of OQ. Larger buffers did provide some reduction in delay, which was significant for very heavy loads only. Moreover, the improvement was diminishing as the size of the fabric was increasing.

Under full permutation traffic, the reduction factor for average delay is almost 2×. With regard to worst-case delay, hierarchical cell distribution yields lower delays, as well. The reduction factor is also increasing with increasing fabric size.

## 5.2 Traffic with Congested Flows

TODO: perform simulations with input traffic that has bursty arrivals and/or hot-spots.

# Chapter 6

# Open Issues and Future Work

Definitely, this is a work-in-progress report. A number of issues still remain to be resolved. These issues generally fall into two categories: regarding parts of the architecture already studied, and regarding areas of future work. Issues regarding cell distribution are presented at the end of chapter 3. Issues regarding the scheduling architecture are presented at the end of chapter 4. We now list the main areas of future work:

- **Buffer Organization**:

  Alternative organizations for the buffers at the concentrator and expander switching elements:

  - which buffers does it make sense to make larger than 1 slot ?
  - which buffers is it possible to drop completely ?
  - what is the impact of each decision on: (i) performance, (ii) impl. complexity (logic/memory ops in each cycle), (iii) impl. cost, (iv) design complexity ?

- **Multicasting**

- **Flow Control Protocol**:

  The flow control protocol interacts with

  - the buffer organization of the switching elements: is it possible to use simple buffer organizations that still perform as good as more complex ones ?
  - robustness of coordinated round-robin: credit based flow control prevents cell loss due to buffer overflow, thus, limiting the need for resynchronizing the coordinated round-robin scheduler only when cells are lost due to electrical noise.

- **Robustness**:

  Lost cells and credits cause problems at two levels:

- Coordinated round-robin: resequencing errors and/or deadlock at the resequencer.
- Flow control protocol: if $B$ cells are lost (where $B$ is the buffer size), then the next cell will never arrive because the sender has no credits.

This is an interesting topic but we consider it as out of the scope of this work. The reader is referred to the work by Lucent and QFC (TODO: citations).

# Bibliography


[1] V. Benes. Optimal Rearrangeable Multistage Connecting Networks. *Bell Systems Technical Journal*, 43(7):1641–1656, July 1964.

[2] 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.

[3] Fabio M. Chiussi, Denis A. Khotimsky, and Santosh Krishnan. Generalized Inverse Multiplexing for Switched ATM Connections. In *IEEE GLOBECOM'98: The Bridge to Global Intergration*, volume 5, pages 3134–3140, November 1998.

[4] Sundar Iyer and Nick McKeown. Making Parallel Packet Switches Practical. In *IEEE INFOCOM*, March 2001.

[5] M. Katevenis. Fast Switching and Fair Control of Congested Flow in Broad-Band Networks. *IEEE Journal on Selected Areas in Communication*, 5(8):1315–1326, October 1987.

[6] M. Katevenis, D. Serpanos, and E. Spyridakis. Switching Fabrics with Internal Backpressure using the ATLAS I Single-Chip ATM Switch. In *GLOBECOM'97*, pages 242–246, November 1997.

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

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

[9] Donpaul Stephens. Implementing Distributed Packet Fair Queueing in a Scalable Switch Architecture. Master's thesis, Carnegie Mellon University, May 1998.

[10] J. Turner. Design of a Broadcast Packet Switching Network. *IEEE Transactions on Communications*, 36(6):734–743, June 1988.

[11] L. G. Valiant and G. J. Brebner. Universal Schemes for Parallel Communication. In *Proc.* $13^{th}$ *ACM STOC*, pages 263–277, 1981.

[12] I. Widjaja and A. Leon-Garcia. The Helical Switch: A Multipath ATM Switch which Preserves Cell Sequence. *IEEE Transactions on Communications*, 42(8):2618–2629, August 1994.