Providing QoS Guarantees in Input Buffered Crossbar Switches with Speedup

by

Anna Charny

B.S., Moscow Institute of Physics and Technology (Russia)
S.M., Kalinin State University (Russia)
S.M., Massachusetts Institute of Technology (1994)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

August 1998

Signature of Author . . . .
Department of Electrical Engineering and Computer Science
June 30, 1999

Certified by . . . .
Dr. David D. Clark
Senior Research Scientist, Laboratory for Computer Science
Thesis Supervisor

Accepted by . . . .
Prof. Arthur C. Smith
Chairman, Department Committee on Graduate Students

Providing QoS Guarantees in Input Buffered Crossbar Switches with Speedup

by

Anna Charny

Submitted to the Department of Electrical Engineering and Computer Science
on June 30, 1999, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

This dissertation investigates a number of issues related to providing Quality of Service guarantees in input-buffered crossbar switches with speedup. It is shown that speedup of 4 is sufficient to ensure 100% asymptotic throughput with any maximal matching algorithm employed by the arbiter. It is also demonstrated that the crossbar architecture is capable of providing delay guarantees comparable to those known for output-buffered switch architecture. Several algorithms which ensure different delay guarantees with different values of speedup are presented and analyzed.

Thesis Supervisor: Dr. David D. Clark
Title: Senior Research Scientist, Laboratory for Computer Science

To my Father

Acknowledgments

I would like to thank my thesis advisor, David Clark, who has given me a perfect combination of guidance, support and independence. His technical comments have always been of great value, and his personal guidance has been an inspiration to me. I thank him for all the help he gave me in writing this thesis. Aside from making it a much more understandable document, he suggested the possibility of a generalization of Theorem 20 in Chapter 5, which led to a more general result of Theorem 29.

I am grateful to my thesis readers Robert Gallager and Balaji Prabhakar for their genuine interest in my work. Robert Gallager has helped me significantly improve the quality of the presentation of my work. His encouragement has been of tremendous importance to me. I thank Balaji Prabhakar for many valuable discussions we had in the course of work on this thesis. His work with Nick McKeown on FIFO output-buffered switch emulation in combined input-output buffered crossbars has been a source of important insight on the problems considered in this thesis. Many of the proofs in this dissertation are similar in spirit to the methodology used in their earlier work.

I thank my colleagues P. Krishna, Naimish Patel and Bob Simcoe at Digital Equipment Corporation for the many stimulating discussions we had while working together on crossbar arbitration issues in the summer of 1997.

I am grateful to Digital Equipment Corporation and Cabletron Systems for the generous support of my graduate studies.

I thank Jon Bennett and Donpaul Stephens for their insightful comments on the earlier drafts of this thesis.

And last but not least, I thank my family for their love, understanding and unyielding support.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation and Background

In the last decade there has been a vast amount of work on providing service guarantees in Integrated Services Networks. The need to support a large variety of applications with diverse quality of service (QoS) requirements such as voice, video-on-demand, real-time video conferencing etc., along with the development of fiber-optics technology, have fueled the need for high-capacity high-speed switching technologies which are capable of providing high-quality QoS. It has been widely accepted that scheduling mechanisms such as Weighted Fair Queuing are needed for providing high-quality QoS. A large number of scheduling mechanisms with various degrees of implementation complexity and QoS guarantees such as [2], [4], [9], [10], [20], [25], [28], etc. have recently been developed. Yet, the vast majority of this work has been done in the context of *output-buffered* switches. In output buffered switches a cell arriving at an input port is immediately forwarded to its output, where it is buffered until it can be transmitted over the appropriate output link. The order of transmission of cells waiting at the output is controlled by a scheduler. The bandwidth and delay guarantees that can be provided depend solely on the properties of this scheduler and the shape of traffic at the entry to the switch. In particular, many of the WFQ-like algorithms such as [2], [4], [20], [25] are capable of ensuring that each packet can be guaranteed a bound on the aggregate delay which is only a function of the flow's own negotiated rate and its own burstiness at the entry to the switch, independently of the behavior or the number of other flows. Other algorithms such as [9], [10] attempt to trade-off implementation complexity for less stringent guarantees (e.g. the delay may depend on the number of flows in the scheduler).

While the output-buffered architecture appears to be especially convenient for providing QoS guarantees, it has a serious limitation: both the switch fabric and the memory at the output of the switch must be capable of running at least at the aggregate speed of all inputs across the switch. Given the current state of technology, at multi-gigabit and terabit speeds it becomes prohibitive to build output-buffered switches. Moreover, the rate at which technologically available memory speeds grow appears to be substantially lagging behind the growth rate of processor speeds. Hence, it is likely that memory speed limitations will remain a bottleneck in the foreseeable future as well.

As a result, many practical high-speed implementations are based on the input-buffered crossbar architecture [18], [29], [30]. In this architecture, buffering occurs at the inputs, and the speed of the memory does not need to exceed the speed of a single input. Given the current state of the art, this architecture is widely considered to be substantially more scalable than output-buffered switches. However, the crossbar architecture itself presents many technical challenges that need to be overcome in order to provide bandwidth and delay guarantees.

The crossbar architecture employs a set of programmable interconnects between the inputs and the outputs. It typically assumes that data is packetized into *cells* of the same size. While this assumption is true for cell-switching networks such as ATM, this is clearly not the case for variable packet size networks such as IP and Ethernet. As a result, if a crossbar switch is used in a variable-size network, packets are typically fragmented into fixed-size cells at the input, and cells are re-assembled into packets at the output. Upon arrival to the switch, cells are buffered at the input, where they wait until they can be transmitted to the output. The cells are dispatched to the outputs in a synchronized manner. That is, time is divided into cell slots. In each cell slot, some cells are transferred from the inputs to the outputs. Unlike an output buffered switch where a cell from each input can be simultaneously transmitted to the same output, the set of cells that can be simultaneously transmitted from inputs to outputs in a crossbar must satisfy the so-called *crossbar constraint:* At any point in time, each output can only accept data from a single input, which must concurrently be transmitting data only to that output. Although under the ideal circumstances it may be possible to transmit a cell from each busy input (or to each busy output) at the same time, the crossbar constraint may substantially reduce achievable throughput. The benefit, however, is that due to the crossbar constraint the memory at the output need not run faster than at the speed of the input, which is clearly an advantage compared to the output-buffered switch, where the output needs to run n times faster (where n is the number of inputs in the switch). The other side of the coin however, is that the crossbar constraint is also the main cause of the difficulty in providing both bandwidth and delay guarantees in this architecture. This issue will become clear from the subsequent discussion.

Recently, a new architectural approach has been suggested in [22], which attempts to overcome this difficulty by eliminating the crossbar constraint and the overhead for fragmentation and re-assembly, while making use of the benefits of the crossbar architecture. In this approach, the crossbar itself contains additional internal buffering with a small amount of buffering per input-output pair. Each input can schedule its cells independently and can put them into the appropriate internal switch buffer without fear of conflicting with other inputs. The outputs, in turn, can pull the cells out of these buffers independently of each other. In order to avoid overflowing these buffers, the switch employs a back-pressure mechanism which stops the input from transmitting new cells until space becomes available there. It is shown that with appropriate scheduling mechanisms at the inputs and the outputs, as well as an appropriate back-pressure mechanism, such a *buffered crossbar* architecture is capable of providing bandwidth and delay guarantees comparable to those available for output-buffered switches.

While this architecture may have some appealing properties, it requires use of a non-standard crossbar and flow control mechanisms integral to the datapath of the crossbar itself. Traditional crossbars have the appeal of being commercially available and are conceptually simpler since no internal buffering or flow control is required internally. As a result, this dissertation concentrates on investigating the means for providing QoS guarantees in a traditional crossbar. The next Section provides additional background on the scheduling algorithms and architectures for traditional crossbars and further motivates the work presented in this dissertation.

1.2 Previous Work Related to Traditional Crossbar Switches.

1.2.1 Providing Bandwidth Guarantees in the Crossbar Architecture.

One of the oldest and perhaps best-publicized results regarding the limitations of an input-queued crossbar switch with respect to its ability to provide any bandwidth guarantees belongs to Karol et.al. [14]. It was shown in this work that for the case when there is a single FIFO queue at each input it is impossible to ensure high throughput to the switch due to the so-called head-of-the-line (HOL) blocking. In particular, for uniform random arrivals of input traffic, the achievable throughput is only 58.6%.

HOL blocking results from a situation in which several inputs have the HOL cell in their FIFO queues destined to the same output. Due to the crossbar constraint, only one of these inputs can send to the output, while the other inputs remain idle even if there are cells in their FIFO (queued behind the HOL cell) that are destined to currently idle outputs. One way of reducing the effect of HOL-blocking is to increase the speed of the switch fabric compared to the speed of the input/output channel. If the switch fabric can run S times faster than an individual input or output, it is said that the switch has a *speedup* of S. There have been a number of studies such as [6],[13] which showed that an input-buffered crossbar with a single FIFO at the input can achieve about 99% throughput under certain assumptions on the input traffic statistics for speedup in the range of $4 - 5$. A more recent simulation study [11] suggested that speedup as low as 2 may be sufficient to obtain performance comparable to that of output-buffered switches. However, all these results are of statistical rather than deterministic nature and are limited by specific assumptions on the arrival distribution.

Another way of eliminating HOL-blocking is by changing the queueing structure at the input. Instead of maintaining a single FIFO at the input, a separate queue for each output can be maintained at each input. The arbitration problem then reduces to that of computing a matching on a bi-partite graph (see, for example [1]). It can be shown that with this queueing structure, even without any speedup, a number of arbitration algorithms, all of which are based on computing a *weighted maximum matching*[1], ensure 100% asymptotic throughput for independent arrivals [16]. However, the complexity of computing the *maximum matching* at each arbitration epoch is quite high. As a result, many of the practical arbitration mechanisms are based on computing a *maximal matching*[2] such as in [1]. While computing a maximal matching is generally substantially easier than computing a maximum matching, maximal matching introduces a problem of its own. It turns out that the simple algorithms such as [1],[24] may no longer achieve 100% throughput, even when no inputs and no outputs are overbooked. The loss of bandwidth can be quite substantial in the absence of speedup. An example of an intuitively fair arbitration mechanism which fails to ensure a 100% bandwidth guarantee, even when the input rates of all flows are such that no input and no output is overbooked, is given

[1]A (weighted) maximum matching on a bipartite graph is defined as a set of edges between input and output vertices with the maximum number of edges (total weight) among all possible sets satisfying the constraint that any vertex in the graph is adjacent to at most one edge in the set.

[2]A maximal matching on a bi-partite graph is defined as a set of edges between input and output vertices such that the constraint that any vertex in the graph is adjacent to at most one edge in the set holds, and no more edges can be added to the set without violating the above constraint. Clearly, any maximum matching is also a maximal matching, but a maximal matching may not be a maximum matching.

in Chapter 2. In fact, to the best of my knowledge no algorithm based on maximal (but not maximum) matching is known to guarantee 100% throughput without speedup for arbitrary input traffic.

However, it has been empirically known that increasing the speedup factor to 2 appears to practically eliminate this problem. It is interesting to note here a potential linkage of this effect to a well-known result from graph theory stating that the size (weight) of any maximal matching is no less than half of the size (weight) of the maximum matching. This result intuitively suggests that at any matching phase an algorithm based on computing a maximal matching would send no less than half of the cells that would be sent if the maximum matching would be computed, implying that with speedup factor of 2 one should be able to send at least as many cells with a maximum matching as is sent with a maximum matching in the absence of the speedup[3]. Despite this intuition and empirical results, it is unclear whether in fact it is true that any maximal matching algorithm would ensure 100% throughput with speedup of 2.

An important breakthrough in understanding the effect of the speedup on the performance of the switch came in [21], where it was shown that there exists a maximal matching algorithm in a crossbar with speedup $S = 4$ that, under arbitrary traffic assumptions, can emulate, or strictly mimic, the behavior of an output-buffered switch with a single FIFO per output. Despite the theoretical importance of this result, the algorithm presented there appears too complex to be feasible in practice. Another algorithm described in [19] has been shown to emulate a FIFO output-buffered switch with speedup $S = 3$ [23]. One simple consequence of these results is that as long as no input and no output is overbooked, it is possible to guarantee a 100% throughput to all flows.

[3]An obvious flaw in this argument is that given identical inputs, with the exception of the first matching epoch, the maximal matching algorithm (with the speedup of 2) and the maximum matching algorithm (without speedup) see different sets of cells (different bipartite graphs) for which the matching is required, and therefore it is unclear how to compare the number of cells sent.

1.2.2 Results Related to Providing Delay Guarantees in Crossbar Switches.

With relation to providing strict delay guarantees for crossbar switches, surprisingly little prior work is known in this context. Perhaps the only known approach to this problem prior to the results of this dissertation is based on the off-line computation of a schedule which is computed at connection setup time [1], [12]. The computation of the schedule is quite complicated and is therefore typically done in software. Since the schedule needs to be recomputed every time a new connection starts up, this approach is basically unsuitable in a dynamically changing environment where new flows can come and go quite frequently. Another limitation of this approach is the necessity to severely limit the supported rate granularity in order to limit the size of the schedule. This leads to lack of flexibility in supporting a range of applications with heterogeneous bandwidth requirements.

Recently, an important result in providing both bandwidth and delay guarantees in a crossbar has been reported in [7] and [26][4], where it was shown that $S = 2$ suffices not only for emulating an output-buffered FIFO switch, but also for emulating an output-buffered switch with a WFQ-like scheduler. This implies that the crossbar architecture is theoretically capable of providing delay guarantees *identical* to the best available for output-buffered switches. However, the complexity of these algorithms appears to be prohibitive for practical real-time high-speed implementations. A more detailed discussion of the relationship of this work to the results of this dissertation is deferred till Chapter 7.

Thus, there still appears to be a need for practical means for providing strict delay guarantees in the crossbar architecture. This dissertation investigates the problem of providing such guarantees.

[4]Unfortunately, there is an error in the algorithm published in [26]. Please refer to http://www.cs.cmu.edu/~istoica/iwqos98-fix.htmp for a discussion of the error.

1.3 Outline

The remainder of this dissertation can be outlined as follows. Chapter 2 is dedicated to investigating the conditions sufficient for providing 100% bandwidth guarantees in crossbars with limited speedup where arbitration is based on a computation of a maximal matching algorithm. It will be shown there that the speedup $S > 4$ is sufficient to ensure that *any* maximal matching algorithm guarantees that each flow achieves 100% of its bandwidth as long as no input and no output is overbooked. It will also be shown that there exist maximal matching algorithms which achieve the same bandwidth guarantees with a lower speedup value. In particular, a maximal matching algorithm providing a 100% bandwidth guarantee with speedup $S = 2$ is described at the end of Chapter 2.

Chapters 3, 4 and 5 are dedicated to investigating several algorithms that ensure deterministic delay guarantees in a crossbar architecture. It is demonstrated that the architecture needed to provide such guarantees can be conceptually subdivided into three relatively independent building blocks: a QoS-capable scheduler employed by the input channels, a QoS-capable scheduler employed by the output channels, and an arbiter which computes a maximal matching between the inputs and the outputs. It is shown that while these three pieces of the architecture can be designed independently of each other, the resulting delay guarantees will strongly depend on the properties of each individual piece in a predictable way. In particular, Chapter 3 is dedicated to describing a system in which the inputs and the outputs run independent QoS-capable schedulers, while the arbiter computes a maximal matching by attempting to give preference to those cells which have been scheduled by the input scheduler earlier. It is shown that as a result of such arbitration policy the worst case delay guarantees depend linearly on the size of the switch. Chapter 4 is dedicated to a discussion on how the properties of the input and output schedulers affect the delay guarantees that can be obtained across the switch. It is demonstrated how the accuracy of both the input and the output schedulers affect the overall guarantees that can be provided. It is further argued that in order to allow for sufficiently tight delay guarantees in this context, the input scheduler should provide a rate-controlled service. In particular it is shown that a rate-controlled version of WF^2Q [4] is well-suited for this purpose because of its high accuracy.

In Chapter 5, it is shown that it is possible to provide deterministic delay guarantees which are independent of the size of the switch. An arbitration mechanism which provides tight delay guarantees with the speedup of $S = 6$ is described in detail.

Chapter 6 is dedicated to a discussion on how to accommodate both flows requiring strict delay guarantees and traffic requiring a lower grade of service in the crossbar architecture. A simple scheduling mechanism is described which provides delay guarantees to delay-sensitive flows while ensuring that "lower grade" traffic fully utilizes the bandwidth unused by guaranteed traffic. It is also shown that mechanisms similar to those described in Chapters 3-5 can be used to ensure that bandwidth is distributed fairly among the "lower grade" flows.

Finally, Chapter 7 summarizes the results of this dissertation and outlines areas for future research.

Chapter 2

Some Delay Properties of Loosely Shaped Traffic in Input-Buffered Crossbars with Limited Speedup

2.1 Preliminaries

We consider an input-buffered nxm crossbar with speedup S. Here n is the number of *input channels*, m is the number of *output channels*. Each input and output channel can have one or more[1] *ports*, each port being connected to an input or output *link*. We make several assumptions on the architecture of the crossbar, all of which are the standard properties of a crossbar switch, with the exception of the requirement that the arbitration algorithm is a maximal matching algorithm[2]:

1. the capacity of each input and each output channel is the same (we denote it by C); for convenience we choose the unit of time such that $C = 1\frac{cells}{unit\ time}$;

[1]The use of several ports per channel is typically motivated by the need of multiplexing several slow links into one high-speed channel.

[2]In practice many simple arbitration algorithms compute only several iterations of a maximal matching, falling short of complete computation.

2. the capacity (speed) of the switch fabric is $S \geq 1$ times greater than the speed of each channel (S is assumed to be a rational number);
3. each input has some buffer structure to hold arriving traffic until it can be dispatched to the outputs[3];
4. each output has some queuing structure to accommodate potentially bursty traffic arriving from the inputs;
5. the arbiter operates in *matching phases;* the duration of a matching phase is $\frac{1}{S}$ units of time; in general no assumptions are made on synchronization between the switch clock and the input (output) channel clock; the beginning of a matching phase will be also referred to as *matching opportunity;*
6. during each matching phase a set of cells is chosen to satisfy the so-called *crossbar constraint:*

   at most one cell can leave any input channel and at most one cell can enter any output channel during a single matching phase;
7. the arbitration policy is *maximal matching* ;
8. those cells which arrived at or before the beginning of a matching phase can be considered for arbitration during this phase.

There could be a number of maximal matching policies. In the context of crossbar arbitration, the standard graph-theoretic definition of maximal matching can be equivalently formulated as follows:

Definition 1 *An arbitration policy is called a* maximal matching *policy iff at the end of any matching phase any cell buffered at any input i destined for any output j remains at*

[3]In principle the buffers can be located at the port level or channel level. For reasons that will shortly become clear it is more convenient to assume that the buffers are at the channel level.

the input if and only if during this matching phase some other cell has been transmitted either from input i and/or to output j.

Definition 2 *The time required to transmit one cell at channel speed is called a* cell slot.

Unless explicitly stated otherwise, a cell slot is used as a unit of time. The unit of rate (speed) is chosen as $\frac{cell}{cell\ slot}$, so that the input/output channel speed is $C = 1\frac{cell}{cell\ slot}$.

2.2 Bandwidth and Delay Guarantees for Arbitrary Maximal Matching Policies.

The main result of this Section is that as long as the incoming rates[4] of all flows[5] are feasible (i.e. no input and no output channel (port) is overbooked in the sense that the sum of rates of all flows sharing the channel (port) does not exceed its capacity), and as long as the combined burstiness of traffic arriving to any input and any output is bounded, then a fixed speedup which is independent of the size of the switch suffices to ensure that *any maximal matching policy* can guarantee that each flow is ensured 100% of its throughput.

To emphasize the significance of this result note that many simple arbitration algorithms computing a maximal matching in the absence of speedup (that is, with $S = 1$) cannot provide a 100% bandwidth guarantee even for feasible rate assignments (even when the arrivals are ideally spaced according to the assigned feasible rates, i.e. with no burstiness at all). Moreover, to the best of my knowledge no results are known about *any maximal matching algorithm* (which is not at the same time a *maximum* matching) which can provide such guarantees with $S = 1$. The following examples show how simple maximal matching algorithms may fail to ensure 100% throughput for feasible rates.

[4]The rate of a flow is to be precisely defined later

[5]A flow is defined as a source-destination pair with an associated transmission rate.

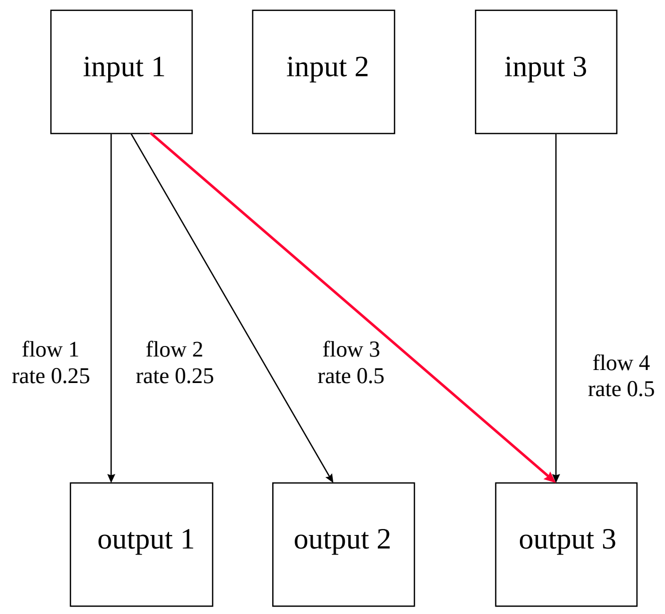

Figure 2-1: Example of Underutilization Caused by a Maximal Matching Algorithm. Flow 3 achieves only 50% of its bandwidth.

Consider a crossbar switch in a cell-switching network with three inputs and three outputs. All inputs and outputs are of unit capacity. There are three flows (denoted f_1, f_2, f_3) arriving at input 1, destined to outputs 1, 2 and 3 respectively. Flow f_4 arrives at input 3 and is destined to output 3. There are no other flows. Flows f_1 and f_2 are assigned rates 0.25, while flows f_3 and f_4 are assigned rates 0.5. Note that such rate assignment is feasible since no input and no output is overbooked. Assume for simplicity that each flow arrives from a separate physical link (i.e. input channel 1 has three input ports) so that the arrivals of all flows are ideally spaced: a cell of flows f_3 and f_4 arrives every second cell slot, and a cell of flows f_1 and f_2 arrives every fourth slot, starting from time zero. This configuration is shown in Figure 2-1.

At each cell slot a maximal matching is computed by the following iterative request/grant protocol. Initially each input chooses one of its flows by some rule and sends a *request* to the appropriate output. When an output receives one or more requests it chooses one of the requests by some rule and sends a *grant* to the appropriate input. If input i receives a grant from output j, the pair (i, j) is added to the matching and is removed from consideration for the duration of the current computation of the maximal matching. It is said in this case that the input i and the output j have been *matched*. The request/grant exchange is repeated until no more matches can be made.

The two examples discussed below differ in the rules by which the inputs and the outputs determine the flow to send request/grant for. In the first example input 1 chooses flows f_1, f_2, f_3 with priorities $1, 2$, and 3 respectively (1 being the highest priority), while output 3 chooses flows f_3 and f_4 with priorities 4 and 3. In this case flow f_3 which should be chosen every other cell slot on the average will in fact be consistently scheduled only every fourth slot, thus losing 50% of its bandwidth.

The loss of bandwidth in this example is not that surprising since flow f_3 is treated at consistently lower priority than its competition at the input and output channels. A lot more surprising is the fact that even intuitively "fair" algorithms can result in a similar loss of bandwidth. To see this consider another example where each input channel uses a round-robin schedule to choose a flow to send a request for. Each output logically maintains an entry for each of the inputs and a round-robin pointer for the inputs. Each input sends a request to the arbiter for the queue currently next in its round-robin schedule (skipping all queues which have no cells in them). If the output receives one or more requests, it sends back a *grant* for the input appearing next in its round-robin schedule (skipping all the inputs for which it has no requests). Once a pair (i, j) has been matched, both input i and output j advance their round-robin pointers and remove themselves from the current computation of the maximal matching. The matching process iterates as described above.

Stepping through this algorithm for the four flows in the configuration with the ideal arrival times as described above one can verify that flow f_3 can achieve only rate 0.25 rather than its input rate 0.5, thus loosing 50% of its bandwidth. This result can be intuitively explained as follows. Due to the contention at output 3 flow f_3 is chosen by its output only 50% of the time since it competes with flow f_4. However, 50% of all the times when f_3 is chosen by its output, it looses the contention with flows f_1 and f_2 at its input.

This example demonstrates that repeated arbitration conflicts in the absence of speedup can cause consistent loss of bandwidth even for an intuitively fair arbitration algorithm.

A possible approach to eliminate the loss of bandwidth due to arbitration conflicts is to use speedup. In fact, it has been empirically known that increasing the speedup to $S = 2$ appears to practically eliminate the loss of bandwidth. However, it has so far been unknown how much speedup is really needed to *ensure* no loss of bandwidth. One of the goals of this Chapter is to investigate the values of speedup sufficient to eliminate this problem for *any* maximal matching arbitration algorithm.

Next, it is necessary to formalize the notions of rate feasibility with limited burstiness. Consider first the case when all flows f are *leaky-bucket constrained* [27] with parameters (r_f, b_f). That is, in any interval $[t_1, t_2)$ the amount of data $A_f(t_1, t_2)$ of a flow f arriving at the switch satisfies

$$A_f(t_1, t_2) \leq r_f(t_2 - t_1) + b_f \tag{2.1}$$

Here r_f is the flow's assigned rate (which is typically negotiated at connection setup), while b_f is the limit on the maximum burstiness of the flow. The conformance of input traffic to a leaky bucket is the standard assumption for the so-called *guaranteed flows*, which are typically required to shape their traffic at the entry to the network. Assuming output-buffered switches with WFQ-like schedulers inside the network, it can be shown that if a flow conforms to a leaky bucket at the entry to the network, it also conforms to a leaky bucket with the same rate (but potentially larger burst size) at the entry to any switch inside the network [20]. As will be shown later, the scheduling algorithms considered in this dissertation for input-buffered crossbar switches also ensure that a flow shaped by a leaky bucket at the entry to the network conforms to a leaky bucket with the same rate but potentially larger burstiness at the entry to any switch inside the network.

In this context *rate feasibility* means that the sum of assigned rates r_f of all flows sharing a particular switch channel (input or output) does not exceed the capacity of this channel. It is easy to see that given a maximum number N of flows supported by the switch and a bound b on individual flow burstiness, the total amount $A_i(t_1, t_2)$ of traffic arriving at the input to the switch in any interval $[t_1, t_2)$ and sharing either any input or any output channel i satisfies

$$A_i(t_1, t_2) \le C(t_2 - t_1) + bN \tag{2.2}$$

In the case of traffic with less stringent QoS requirements, such as *best effort*, traffic may not be shaped at the network entry, and the notion of an assigned rate may not be meaningful. For this type of traffic congestion control algorithms are typically used to ensure that data is not lost due to buffer overload. The degree with which this goal is achieved varies greatly with the type of congestion control algorithm utilized in the network. It is assumed however that the congestion control algorithm is "good enough", which means that, given a sufficient buffer size, no data is lost. This can be formalized by assuming that there exists some (potentially large but finite) value B which is *independent of time,* such that input traffic sharing any input (or any output) i in the switch satisfies the constraint

$$A_i(t_1, t_2) \le C(t_2 - t_1) + B \tag{2.3}$$

for any interval $[t_1, t_2)$. The value B in this case determines the buffer requirements of the congestion control algorithm.

Note that (2.3) does not imply that an *individual* flow conforms to a particular rate (such as the rate r_f in the case of leaky-bucket constrained flows). Instead, it is only required that the *combined rate* of all flows sharing a particular channel does not exceed the capacity of that channel by more than a fixed bound B in any interval of time. That is, the condition (2.2) is a special (stronger) case of (2.3). This Chapter assumes only that (2.3) is satisfied, without requiring shaping the traffic of any individual flow.

The main result of this Chapter is the fact that for any speedup $S > 4$ any maximal matching arbitration policy ensures 100% throughput for traffic satisfying assumption (2.3).

This fact is proved by showing that each cell is delivered to its output within a certain fixed bound after its arrival to the switch. This immediately implies that the rate at which any flow enters its output is asymptotically the same as its input rate to the switch. Assuming that the output channel employs any of the known schedulers which preserve the asymptotic rates of the flows in the context of an output-buffered switches (such as FIFO, WFQ, etc.), it follows that a flow is guaranteed its bandwidth across the switch.

The following Theorem states that each cell is delivered to the output within a certain bound after its arrival:

Theorem 3 *If input traffic satisfies assumption (2.3), then, given a sufficiently large buffer size, for any speedup $S > 4$ and any maximal matching arbitration policy a cell arriving to the input at time t will be delivered to the output by time $t + \frac{2B-1}{S-4} + \frac{1}{S}$.*

First, the following simple Lemma will be proved:

Lemma 4 *The number of matching opportunities $M(t, t+\tau)$ occurring inside any interval $[t, t+\tau)$ satisfies*

$$\tau S - 1 \leq M(t, t+\tau) \tag{2.4}$$

Proof of Lemma 4.

The Lemma follows immediately from the observation that at least $\lfloor \tau S \rfloor - 1$ matching phases (each of length $\frac{1}{S}$ units of time) must be fully contained in an interval of length τ. This implies that the number of matching opportunities (which correspond to the *boundaries* of matching phases) inside any interval of length τS must be at least $\lfloor \tau S \rfloor \geq \tau S - 1$. ■

Proof of Theorem 3.

It will now be shown that a cell arriving at time t will be chosen by the arbiter no later than by time $t + D$. Since it takes exactly time $\frac{1}{S}$ to deliver a cell from the input to the output at the speed of the switch fabric, the statement of the Theorem follows. Suppose that there exists a cell for which the *arbitration delay bound* D is violated. Pick a cell c with the earliest arrival time of all such "violating" cells (breaking ties arbitrarily). Let t denote the time of its arrival. Then, since c violated its delay bound D, and since cells are scheduled only at the discrete cell-time boundaries, it must be that there exists an $\varepsilon_0 > 0$ such that for any $0 < \varepsilon \leq \varepsilon_0$ the cell c is still at its input at time $t + D + \varepsilon$. Since the policy is a maximal matching, in order for c not to be scheduled at or before time $t + D$, it must be that another cell from input i and/or destined to output j was chosen by the arbiter at every arbitration epoch in the interval $[t, t+D]$. Therefore, by Lemma 4 it must be that at least $DS-1$ other cells either from input i or/and destined to output j were chosen by the arbiter in the interval $[t, t+D]$. It will be shown now that there could not be as many cells available to the arbiter in this interval. Note that, by assumption, c is the first cell to miss its deadline D, and therefore all the cells which arrived before time $t - D$ must have already been scheduled by time t. Therefore, each cell that can still be at the input at time t must have arrived in the interval $[t - D, t]$. Therefore, each competing cell that could be available to the arbiter in the interval $[t, t + D]$ must have arrived in the interval $[t - D, t + D]$ and must either share input i or output j with c. From (2.3), the number of such cells arriving in $[t - D, t + D] \subset [t - D, t + D + \varepsilon)$ can be at most $4D + 2B + 2\varepsilon$. This includes c itself, counted both at its input and its output. Therefore the total number of cells competing with c might have in the interval $[t - D, t + D]$ is at most $4D + 2\varepsilon + 2B - 2$. If the number of cells which can possibly prevent c from being scheduled at or before time $t + D$ is less than the minimal number of arbitration opportunities in the interval $[t, t + D + \varepsilon)$, then no cell can be delayed at the input by more than D. Hence, if the inequality

$$4D + 2\varepsilon + 2B - 2 < DS - 1 \tag{2.5}$$

is satisfied, then cell c cannot remain at the input at time $t + D$. Rewriting (2.5) as

$$D(S - 4) > 2B - 1 + 2\varepsilon$$

and choosing ε sufficiently small, and $S > 4$, it can be easily seen that (2.5) is satisfied if $D > \frac{2B-1}{S-4}$. Hence the cell c cannot remain at the input longer than $\frac{2B-1}{S-4}$ after it has been scheduled by its input scheduler. ■

2.3 Reducing the Speedup to S>2

The result of the previous Section is that $S > 4$ is sufficient to ensure each flow 100% bandwidth guarantee (with bounded per-cell delay) with any maximal matching algorithm. However, it is not the case that the condition $S > 4$ is necessary for any maximal matching algorithm to provide such guarantee. It will now be shown that a *particular* maximal matching policy can give the same arbitration delay bound (and hence 100% bandwidth guarantee) with any $S > 2$. The idea here is quite simple - to disallow cells which arrive later to compete with earlier arrivals, thus reducing the amount of potential "competition" and therefore the speedup required to transfer all possible "competition". To do so, consider the following algorithm:

Oldest Cell First maximal matching (OCF): Upon arrival each cell is "stamped" with the time of its arrival. At each matching phase the arbiter does the following: It chooses the cell with the oldest stamp and adds its input and output to the match. It then removes from consideration all cells with the same input and/or output and repeats the previous step until no more cells can be added.

Theorem 5 *If the input traffic satisfies assumption (2.3), then the OCF arbitration policy in a crossbar with speedup $S > 2$ ensures that any cell arriving at time t will be delivered to its output no later than by time $t + \frac{2B-1}{S-2} + \frac{1}{S}$.*

The reduction in the required speedup is enabled by the fact that due to arbitration based on OCF, cells arriving after some cell c cannot be c's competition, because they have a later timestamp. The proof given below is conceptually very similar to that of Theorem 3.

Proof.

Just as in the previous Section it will be shown that any cell will be scheduled no later than $D = \frac{2B-1}{S-2}$ after its arrival. Suppose it is not the case. Then there must exist at least one cell that arrived at some time ξ and was not scheduled by time $\xi + D$. Let t be the earliest arrival time of all such cells and let c denote a cell which arrived at time t and remained at the input at time $t+D$. Since cells are scheduled by the input scheduler only at discrete cell slot boundaries, it must be that there exists some $\varepsilon_0 > 0$ such that for any $0 < \varepsilon < \varepsilon_0$ the cell c is still at the input at time $t + \varepsilon$. Just as in the proof of Theorem 3 it will be shown that this cannot happen because there are not enough "competing" cells to prevent c from being scheduled at or before time $t+D$. The crucial observation here is that by the operation of the OCF algorithm a cell with timestamp t which is waiting at input i and is destined to output j is not scheduled at a matching phase boundary iff a cell *with a smaller or equal timestamp* is scheduled either from input i or/and from some other input destined to output j. Therefore, no cells arriving *after time* t can prevent c from being scheduled (since their timestamps are greater than that of c). As a result, the only cells that constitute c's "competition" are those which arrived at $\tau \leq t$. Since it was assumed that c is the first cell to miss its scheduling deadline of D, it must be the case that any cell arriving before $t - D$ is already scheduled by time t. Hence, the only "competition" of c are those cells which arrived to input i or/and to output j in the interval $[t - D, t]$. By the traffic assumption 2.3 at most $D + B+ \varepsilon$ cells can arrive at any input i, and at most $D + B + \varepsilon$ of cells can arrive at all inputs destined to output j in any interval $[t - D, t] \subset [t - D, t + \varepsilon)$. This includes c itself, counted both at its input and its output. Therefore, the total amount of c's competition is bounded by $2D + 2B + 2\varepsilon - 2$.

By Lemma 4 there are at least $DS - 1$ matching phase boundaries in the interval $[t, t + D]$. In order for c to remain at its input at time $t + D$, at least one competing cell must be scheduled at every matching phase boundary in the interval $[t, t + D]$, and so it must be that $2D + 2B + 2\varepsilon - 2 \geq DS - 1$. However, it is easy to see that for $S > 2$ and sufficiently small ε, for any $D > \frac{2B-1}{S-2}$ in fact $2D + 2B - 2 < DS - 1$. This contradiction shows that as long as $S > 2$, any cell is scheduled by the arbiter no later than time $\frac{2B-1}{S-2}$ after its arrival to the input. Noting that it takes exactly $\frac{1}{S}$ units of time to transmit the cell across the switch completes the proof of this Theorem. ■

Corollary 6 *For any speedup $S = 2 + \varepsilon, \varepsilon > 0$ the matching arbitration policy based on* OCF *ensures 100% bandwidth guarantees to all flows, as long as traffic satisfies the assumption (2.3).*

It can be easily seen that the Theorem holds also if the "timestamp" is any monotonically increasing function of arrival time.

Chapter 3

Delay Guarantees for Per-Flow Shaped Traffic

3.1 The Necessity of Flow Isolation

In the previous Section the bounds on the delay incurred by any cell from the time of its arrival at an input port of the switch to the time it is delivered to its output port were obtained as a function of the speedup and the maximum burst size B that could be experienced by the totality of all flows at a given input or output. The presence of a single misbehaved flow can make B unacceptably large, affecting delays of "well-behaved" flows. Even when each individual flow is well behaved, i.e. if the degree of the individual flow burstiness is relatively small, the combined burstiness of all flows multiplexed to a particular output or input can be quite large. At the input the cumulative burstiness can be quite large especially in the case when the total capacity of all incoming ports at an input exceeds the input channel capacity[1]. Since the total capacity of all input links is typically much larger than that of a single output link, the combined burstiness at the

[1]The channel capacity can be overbooked, for example, if several links multiplexed into a single channel are leased to different users which are not expected to simultaneously utilize the entire bandwidth leased to them. In this case overbooking of channel capacity may be desired to achieve higher channel utilization due to statistical multiplexing.

output can be quite large even when the combined input link bandwidth does not exceed the input channel capacity. There is an obvious necessity to protect well-behaved flows from misbehaved ones, and in general to minimize the effect of the cumulative burstiness on the performance of individual flows.

The issue of providing such flow isolation has been widely discussed in the literature in the context of providing QoS in output-buffered switches. Typical approaches to achieve such isolation include per-flow buffering and a QoS- capable scheduler employing a WFQ-like service discipline such as in [2], [20], etc. This Section will discuss how to achieve the goal of flow isolation and consequently deterministic delay bounds for individual flows which are independent of the burstiness of other flows in the context of input-buffered crossbars.

3.2 Achieving Per-cell Delay Bounds with Flow Isolation.

3.2.1 The Basic Idea

As is well understood from the existing work on providing QoS in a network of output-buffered or shared memory switches, provisioning per-flow queueing and a QoS-capable scheduler at all queuing points in the network can yield end-to-end QoS guarantees for all well-behaved flows. It appears only natural that this is also true for input-buffered crossbars. As can be easily seen, in an input-buffered crossbar with speedup there are in fact two queuing points - at the input and at the output. An immediate extension of the previous QoS work would be to provide per-flow queues both at the input and at the output. It is less clear how to do the scheduling to ensure that arbitration conflicts caused by input/output contention have minimal effect on the resulting QoS guarantees that can be ensured for all flows. It turns out that to ensure that the delays experienced by an individual flow are independent of the combined burstiness, a policy conceptually similar to the Oldest Cell First policy described in the previous Chapter can be employed in combination with techniques developed for providing QoS in output-buffered switches.

The main idea here is to modify the "arrivals" of individual flows by passing them through a QoS-capable rate-controlled scheduler. Only cells already "filtered through" a rate-controller are considered conceptually "arrived" to the switch and are therefore eligible for arbitration. When a cell "emerges" from such a rate-controlled scheduler, it is immediately assigned a timestamp. The arbiter uses the "oldest timestamp first" policy, which is just *OCF* discussed in Section 2.3 in which the actual arrival times are replaced by the times a cell is released from the input rate controller. In the remainder of this Section this will be described in more detail.

3.2.2 The Queueing Structure at the Input

At each input there are per-flow queues. Each queue has a rate associated with it, which is typically assigned to the flow at connection setup time. In a cell-switching network, when cells arrive to the switch, they are simply placed into the corresponding queues. In a packet-switching network, arriving packets are fragmented into cells[2], and the cells are added to the corresponding per-flow queue, where they wait until they can be transmitted to the output channel. In addition to queues per flow, there are also per output queues at each input - one per output. According to a standard convention they are referred to as *virtual output queues.* Unlike the flow queues which contain cells, the virtual output queues contain pointers to cells as will be explained below. Denote by q_f the queue corresponding to flow f. Further, denote by Q_{ij} the virtual output queue at

[2]There are several possible ways of performing fragmentation if the packet size is not a multiple of a cell size. One is generally referred to as *padding*, when the last cell is padded by some special "dummy" symbols. While being the simplest, this method has a disadvantage of wasting potentially as much as half the switch bandwidth (if the size of all packets is a small ϵ longer than the cell size). In this case one can use what is usually referred as *"cell-stuffing".* In this method data from the beginning of the next packet is attached to the end of the previous packet. While this approach may be beneficial in certain cases, it introduces additional complexity in performing fragmentation and reassembly. Moreover, the benefit in saved throughput depends on the assumptions about the packet length distribution, which are not always known. This thesis will assume a simple fragmentation with padding, in most cases ignoring a potential loss of bandwidth that may be associated with it.

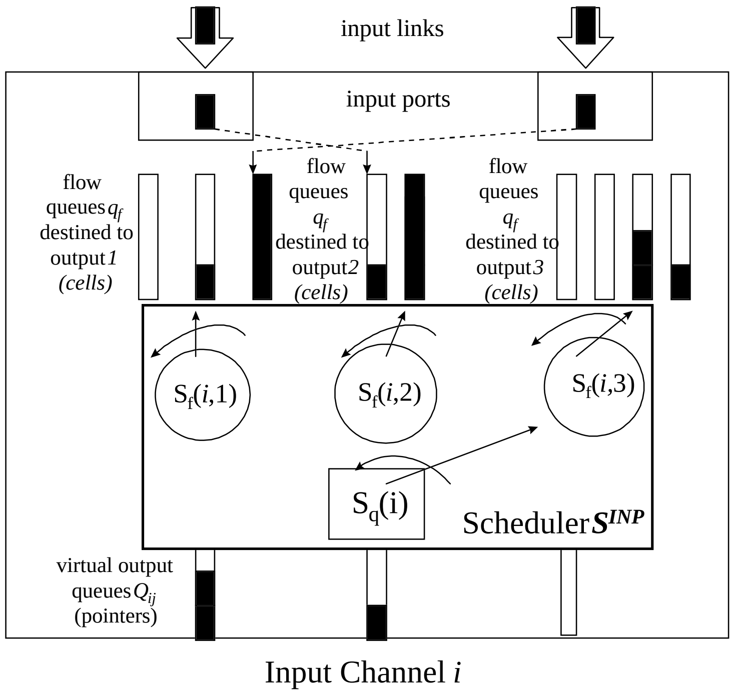
input links
input ports
flow queues q_f destined to output 1 (cells)
flow queues q_f destined to output 2 (cells)
flow queues q_f destined to output 3 (cells)
$S_f(i,1)$
$S_f(i,2)$
$S_f(i,3)$
$S_q(i)$
Scheduler S^{INP}
virtual output queues Q_{ij} (pointers)
Input Channel i

Figure 3-1: Buffering and scheduling structure at the input channel. This example corresponds to a switch with 3 output channels.

input i corresponding to output j. Each queue q_f is assigned a rate r_f corresponding to the rate assigned to flow f. Assume that the per-flow queues at each input are grouped according to the destination output, so that each group of flow queues is associated with a particular virtual output queue. Each virtual output queue Q_{ij} is also assigned a rate R_{ij} which is the sum of the assigned rates of all flows at the input destined for the output associated with this queue, i.e. $R_{ij} = \sum_{f \to j} r_f$, where $f \to j$ denotes "f is destined to j". The scheduling mechanism that determines the order of transmission of cells across the switch to their destination output channel is described in the next two Sections. The queuing structure for a single input is shown in Figure 3-1.

Frequently, it will be convenient to imagine that at time zero the flow queues are infinitely backlogged with imaginary "dummy" cells. When a real cell arrives at the input, it replaces the closest dummy cell to the head of its flow's queue. The scheduling mechanisms described below do not distinguish between dummy and real cells. When a dummy cell is chosen to be transmitted to the output channel (as described in the Sections below), it is simply removed from the flow queue.

3.2.3 The Input Rate-Controlled Scheduler

The goal of a rate-controlled scheduler located at an input channel, which will be denoted by $\mathbb{S}^{INP}$, is to ensure that the cells of each flow f become available for arbitration with the frequency corresponding to the flow's rate $r_{f.}$. $\mathbb{S}^{INP}$ is constructed in a hierarchical manner - the top level of the hierarchy ensures that each virtual output queue Q_{ij} is chosen with the frequency R_{ij} corresponding to the combined rate of all flows in the group corresponding to this virtual output queue. More specifically, the top level of the hierarchy at input channel i consists of a rate-controller, which is denoted by $\mathbb{S}_q(i)$. Every channel cell time this rate-controller $\mathbb{S}_q(i)$ chooses a particular Q_{ij}.

At the second level of the hierarchy there are flow-level rate-controllers at each input channel, one for each group of flows at each input i corresponding to a given output j, denoted by $\mathbb{S}_f(i, j)$. The goal of the second level of the hierarchy is to ensure that each flow within the group is chosen with the frequency corresponding to the flow's rate. When Q_{ij} is chosen by $\mathbb{S}_q(i)$, the $\mathbb{S}_f(i, j)$ corresponding to the chosen j is invoked to choose a particular flow f. At this time a pointer to the next so far unscheduled[3] cell of the chosen flow f is added to the tail of the virtual output queue Q_{ij}. The location of schedulers S_q and $\mathbb{S}_f$ with respect to the queueing structure at the input is shown in Figure 3-1.

The time instance when $\mathbb{S}_q(i)$ chooses queue Q_{ij} will be referred to as the *scheduling opportunity* of the virtual output queue Q_{ij}. When a pointer to a cell is added to the appropriate virtual output queue, this *cell is said to be scheduled by the scheduler* $\mathbb{S}^{INP}$. It is important to understand that a cell which has been scheduled by $\mathbb{S}^{INP}$ may not

[3]That is, the cell in the flow queue for which no pointer has yet been added to the corresponding virtual output queue.

be transferred to its output immediately, but may remain in its queue until the arbiter chooses it, as described in the next Section.

In general, a variety of rate-controllers can be used for schedulers $\mathbb{S}_q$ and $\mathbb{S}_f$ at the two levels of the hierarchical scheduler $\mathbb{S}^{INP}$. As will be shown in the Sections below the properties of these schedulers determine the delay a cell can experience at the input channel.

3.2.4 The Arbiter

This Section describes how the cells are removed from the input channel. The scheduler responsible for this removal is called the *arbiter*. At each matching opportunity the arbiter computes a maximal matching among the cells pointed to by the HOL pointers in the virtual output queues. Once a virtual output queue is added to the current matching, the HOL cell of the flow queue q_f pointed by the HOL pointer in this virtual output queue is transmitted to the output channel, at which point this cell is removed from the flow queue, and the pointer to it is removed from the virtual output queue. The goal of the arbiter is to ensure that each virtual queue Q_{ij} is polled (added to the matching) with the frequency corresponding to its rate R_{ij}.

To achieve this goal, the arbiter uses a variant of the OCF algorithm described in the previous Chapter. Conceptually, imagine that any time a pointer to a flow queue is added to a virtual output queue, a timestamp corresponding to the time of this event is stored along with the pointer. At each arbitration epoch, the arbiter computes the maximal matching among all virtual output queues by choosing the oldest timestamp at the head of all virtual output queues at all inputs using these timestamps, as in the OCF algorithm described in the previous Chapter.

However, such an approach presents the following implementation difficulty. Since the arbiter needs to have access to the HOL timestamps of all virtual output queues at each matching phase, storing the timestamps in the virtual output queues would require communicating nxm timestamps from the input channel to the arbiter every matching phase. It turns out that one can substantially reduce the communication overhead by storing the timestamps directly in the arbiter. In this implementation, the arbiter maintains a queue a_{ij} corresponding to each input-output channel pair. Initially all queues are empty. The queues a_{ij} are filled with *timestamps*, corresponding to the time at which the input scheduler $\mathbb{S}_q(i)$ at input i chooses the corresponding Q_{ij}. Recall that the input schedulers operate at channel speeds, and therefore every channel cell slot the input needs to communicate the appropriate information to the arbiter. One way is to straightforwardly transmit the timestamp with the queue index to the arbiter. However, it is easy to see that it suffices to transmit the index alone (which would typically require a smaller number of bits), since the arbiter can simply add the current time to the queue a_{ij} corresponding to this index. Thus, the addition of timestamps to the arbiter queue occurs once per channel cell time, when one timestamp must be added per input channel. The removal of timestamps occurs once per matching phase as follows. At each matching opportunity the arbiter computes a maximal matching as described below. It chooses a_{ij} with the oldest HOL timestamp, adds the corresponding Q_{ij} to the match, and removes the HOL timestamp from a_{ij}. It then removes from consideration all queues with the same input and/or output and repeats the previous step until no more queues can be added to the matching. Once the matching is complete, the arbiter notifies the inputs which Q_{ij} are chosen in the current matching. The HOL cells in the virtual output queues corresponding to this matching are then transmitted to the output channels.

The time instance when the arbiter chooses a particular virtual output queue is referred to as the *arbitration opportunity of this virtual output queue*, or, equivalently, the *arbitration opportunity of the input/output channel pair* corresponding to the virtual queue. The arbitration epoch at which a cell (potentially dummy) of some flow is chosen for transmission to its output channel is referred to as the *arbitration opportunity of this flow.*

3.2.5 The Output Channel Architecture.

Assume that there may be several physical links outgoing from a particular output channel, with an output port associated with each link. In a cell-switching network such as ATM, where no fragmentation and reassembly function is required, a cell arriving from an input to the output channel is immediately placed in the appropriate buffer at the port corresponding to its destination link. In the case of packet-switching networks, assume that upon arrival to the output, cells are placed in *re-assembly queues*. Once all cells of a packet have been collected in the queue, the packet is assembled and placed in the corresponding packet queue at the destination port.

Assume that each output port p employs per-flow buffering and a QoS-capable scheduler $\mathbb{S}_o(p)$. There is substantial flexibility in the choice of a particular $\mathbb{S}_o(p)-$ one can choose from a large arsenal of QoS-capable schedulers available for output-buffered architectures such as [2], [20], etc. The choice of a scheduler defines the portion of delay corresponding to a cell/packet residence at the output. The architecture of the output channel is shown in Figure 3-2.

Finally, the entire switch architecture described in this Chapter is shown in Figure 3-3.

3.3 Considerations for Choosing Input And Output Schedulers

The total delay of a cell in a switch is the combined delay it experiences in the input channel and the output channel. The remainder of this Chapter will examine the dependence of the input and output delay components on the properties of the input and the output schedulers for the case of timestamp-based arbitration described above.

Recall that in the algorithm considered in this Chapter the input delay of a cell has two logical components. The first component, which will be referred to as the *input*

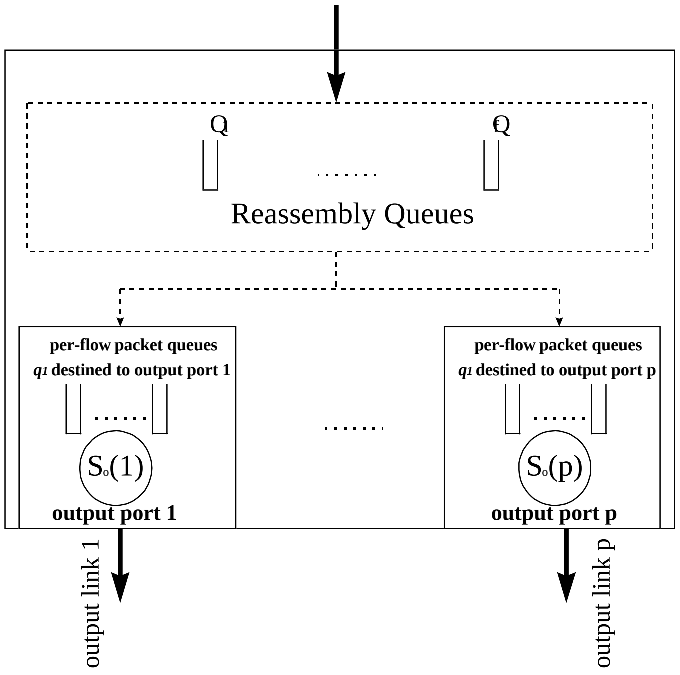

Figure 3-2: Ouput channel architecture. Reassembly queues (indicated by a dotted line boundary) are needed in packet networks but should be omitted in cell-switching networks.

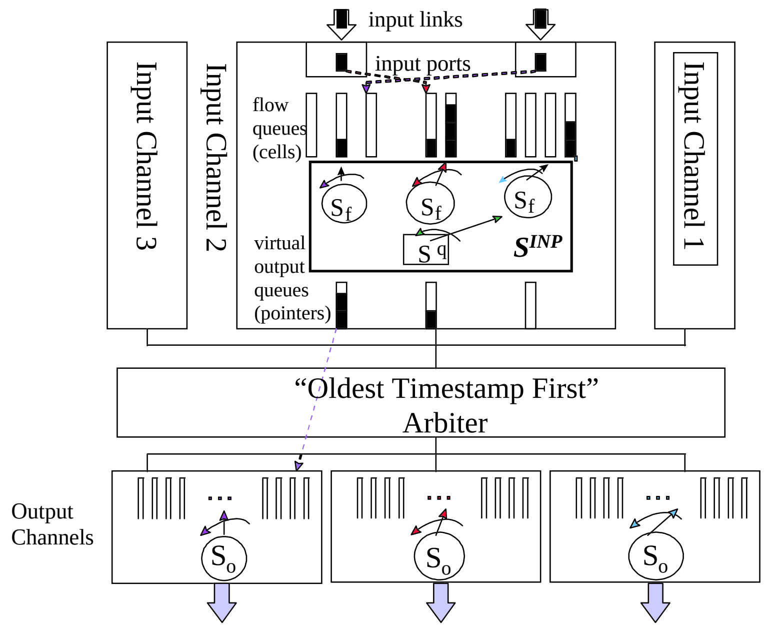

Figure 3-3: The Switch Architecture For Time-Stamp Based Arbitration. A 3x3 switch with a single port per output channel is shown.

scheduling delay in $\mathbb{S}^{INP}$, is the time elapsed from the arrival of a cell at the input channel and the time a pointer to this cell is added to the appropriate virtual output queue. The second component, which will be referred to as the *arbitration delay,* is the time elapsed from the moment the pointer to the cell is added to the tail of the virtual output queue and the time cell is removed from the flow queue and transmitted to the output.

It turns out that both of these components strongly depend on the properties of a particular rate-controlled scheduler $\mathbb{S}^{INP}$ employed at the input, and more specifically on the accuracy with which the rate-controller approximates the ideal "fluid" scheduler.

Intuitively, the quality of a generic rate-controller can be described as follows. Consider some rate-controller operating at unit speed on n flows f with rates r_f satisfying $\sum_f r_f \leq 1$. If cells were infinitely divisible, it would be possible to serve each flow f so that in any interval of length t each flow f would be offered exactly tr_f units service. Such idealized service is referred to as the "ideal" or "fluid" service at rate r_f. However, due to the discrete nature of the system, the amount of service which can be actually offered to a flow will differ from the ideal service. The bound on discrepancy between the ideal and the actual service offered to a flow by the scheduler can be viewed as a measure of the quality of the rate-controller.

To quantify this notion, denote by $A_f(t_1, t_2)$ the amount of actual service offered to flow f in the time interval $[t_1, t_2)$, where the time is measured in channel cell slots, and the amount of service is measured in cells. Suppose that for all flows f the rate-controller ensures that the following constraints are satisfied for all times t_1, t_2:

$$A_f(t_1, t_2) \leq (t_2 - t_1)r_f + \overline{E} \tag{3.1}$$

$$A_f(t_1, t_2) \geq (t_2 - t_1)r_f - \underline{E} \tag{3.2}$$

where $\underline{E} \geq 0$ and $\overline{E} \geq 0$ are independent of time and the flow f. The bounds $\underline{E}$ and $\overline{E}$ can be viewed as the measure of the *accuracy* of the rate-controller: the smaller these bounds, the better the rate-controller. It is easy to see that the meaning of $\underline{E}$ and $\overline{E}$ is how much more or less service (in units of work such as cells or bits) the rate-controller gives a flow compared to its ideal fluid service at its rate. There are several known rate-controllers satisfying (3.1) and (3.2) with different values of $\underline{E}$ and $\overline{E}$. Some of these schedulers will be discussed in the next Chapter.

Equivalently, one can also look at the accuracy of a rate-controller from the point of view of *time discrepancy* rather than of *service discrepancy.* Consider a sequence of cells of a flow f with assigned rate $r_{f.}$ If the flow were serviced as a fluid, starting from time zero, the k-th cell ($k \geq 0$) would start service exactly at time $\frac{k}{r_f}$ (provided of course the cell is there to be served). The ideal rate-controller would give the flow a service opportunity exactly every $\frac{1}{r_f}$ units of time, so that the k-th service opportunity is offered to the flow exactly at time $t_k = \frac{k}{r_f}$ (regardless whether a cell of a flow is actually there or not). If the real rate-controller offers its scheduling opportunities at some times τ_k, the time discrepancy can be defined simply as the difference between the ideal and the actual time of a service opportunity $t_k - \tau_k$. The accuracy of a rate-controller can now be characterized as a bound on this time difference.

While the time and service discrepancy can be used interchangeably, in most cases it will be more convenient to use the service discrepancy bounds (3.1) and (3.2).

It can be intuitively expected that the bound on the delay a cell can incur in the input rate-controlled scheduler depends on the lower bound on service discrepancy $\underline{E}$. Specifically, it is shown in the next Section that the input scheduling delay of any flow depends on the value $\underline{E}$ of the two-level hierarchical scheduler $\mathbb{S}^{INP}$, as well as the individual burstiness of the flow at the input to the switch. Further, in Section 3.3.2 a less intuitive result is proved: the bound on arbitration delay depends on the value $\overline{E}$ of the *top level rate-controller* $\mathbb{S}_q$ in the scheduler $\mathbb{S}^{INP}$. To distinguish service discrepancy bounds of the scheduler $\mathbb{S}_q$ from those of the two-level scheduler $\mathbb{S}^{INP}$, denote the bounds corresponding to $\mathbb{S}_q$ by $\underline{E}^{IQ}$ and $\overline{E}^{IQ}$, and the bounds $\underline{E}$ and $\overline{E}$ corresponding to $\mathbb{S}^{INP}$by $\underline{E}^{INP}$ and $\overline{E}^{INP}$ respectively. It will be shown that in the framework described in this

Chapter the arbitration delay bound is also a function of the number of channels in the switch.

It should be clear that once a cell has entered the output channel, its delay depends on the properties of the output scheduler. It is assumed that the output scheduler is some QoS-capable scheduler, for example one of the WFQ-like schedulers such as PGPS [20]. For these schedulers it is known that the delay bound depends only on the assigned rate (weight) of a flow and its individual burstiness. It is shown in Section 3.3.3 that the bound on the burstiness of any flow *at the entry to the output channel* is determined by the accuracy of the *input* rate-controller.

While it is possible to employ both work-conserving and non-work-conserving schedulers at the output, the case considered in this Section is when the output scheduler is a non-work-conserving rate-controller satisfying (3.1) and (3.2) with some values of $\underline{E}^{Out}$ and $\overline{E}^{Out}$. The case of work-conserving schedulers at the output will be considered later. The bounds on the output delay are derived in Section 3.3.3 below.

Finally, the bound on the total switch delay is derived in Section 3.3.4, where traffic burstiness at the output of the switch is also described.

3.3.1 A Bound on the Input Scheduling Delay

Consider a generic scheduler satisfying (3.2) operating at unit speed on some flows with rate assignments satisfying $\sum_f r_f \leq 1$. The next Theorem gives an upper bound on the delay any cell can experience in such a scheduler. As can be expected, this bound depends on $\underline{E}$ (how much *behind* the ideal service a flow can be in the scheduler) and the degree of burstiness of this flow at the input to the scheduler. Although this result is rather straightforward, it will be repeatedly used in this thesis and therefore is stated as a separate theorem.

Theorem 7 *If flow f is constrained by a leaky bucket with parameters (r, b) at the input to a scheduler satisfying (3.2), then each cell is scheduled no later than time $\frac{b+\underline{E}}{r}$ after its arrival.*

Proof.

Consider some cell c arriving at some time t to the queue of flow f . Suppose first that c starts a busy period of this flow. Then c is scheduled at the first scheduling opportunity of flow f occurring at or after time t. By (3.2) for any $\varepsilon > 0$ the scheduler must start service of at least one cell in the interval $[t, t + \frac{\underline{E}+\varepsilon}{r})$, which implies that c will be scheduled no later than at time $t + \frac{\underline{E}+\varepsilon}{r}$. Since ε can be chosen arbitrarily small, this implies the statement of the Theorem.

Suppose now that c is not the first cell in a busy period of flow f, and let t_0 be the beginning of the busy period containing t. Consider any time $\tau \geq t$ at which c still remains in the queue. By (3.2) in the interval $[t_0, \tau] \supset [t_0, \tau)$ the flow f must have received at least $(\tau - t_0)r - \underline{E}$ scheduling opportunities (in units of service). Since the queue of f has been continuously non-empty in the interval $[t_0, \tau)$, at least $(t_0 - \tau)r - \underline{E}$ cells have been served. Note that all of these cells must have arrived to the queue of f prior to the arrival of c at time t. Since the flow is constrained by a leaky-bucket (r, b), there could have been at most $(t - t_0)r + b$ cells (including c) which could arrive before c in the considered busy period. Therefore, it must be that $(t - t_0)r + b \geq (\tau - t_0)r - \underline{E}$, which implies that for any time τ at which c could still be in the queue $\tau \leq t + \frac{\underline{E}+b}{r}$. Therefore, the waiting time in queue is bounded by $\frac{\underline{E}+b}{r}$. ■

Theorem 7 immediately implies that in the context of the input scheduler $\mathbb{S}^{INP}$, given the leaky bucket parameters (b, r) at the input to the switch (which is assumed independent of the internal switch architecture), the input scheduling delay is fully determined by the bound $\underline{E}^{INP}$ of the scheduler $\mathbb{S}^{INP}$. That is, denoting the input scheduling delay by D^{INP}, Theorem 7 yields

$$D^{INP} \leq \frac{b + \underline{E}^{INP}}{r} \tag{3.3}$$

Note that in addition to giving a bound on the input delay, Theorem 7 also gives the bound on the *output* delay as a function of $\underline{E}^{Out}$ and the burstiness of the flow at the entry to the output channel. However, the output delay depends on the leaky-bucket parameters of the flow's traffic as it enters the output channel. The latter, as will be shown, depends on the properties of the *input* rate controlled scheduler and the *arbitration policy.* A more detailed discussion of this issue is delayed till Section 3.3.3.

3.3.2 A Bound on Arbitration Delay (The Case of Speedup S>2).

In this Section bounds on the delay incurred by each cell which are due to crossbar *arbitration conflicts are derived.* This delay is specifically due to the constraint of the crossbar architecture. The ability to limit it in a timely manner is vitally important to ensure strict bandwidth and delay guarantees. It will be shown here that for the simple arbitration algorithm described in Section 3.2.4 above it is possible to ensure that each cell is delivered to its output within a certain bound from the time this cell is released by the input rate-controlled scheduler. As will be shown, this bound depends on the size of the switch and the speedup factor. More importantly, it will be shown that this delay also depends on the properties of the input rate-controlled scheduler $\mathbb{S}^{INP}$ described in Section 3.2.3, and more specifically on the properties of the top-level scheduler $\mathbb{S}_q$ in $\mathbb{S}^{INP}$.

The next Theorem gives the bound on the arbitration delay as a function of the bound $\overline{E}^{IQ}$ of the top-level scheduler $\mathbb{S}_q$ at the input. The bound also depends on the speedup of the switch fabric and the size of the switch.

Theorem 8 *If the rate-controllers $\mathbb{S}_q$ employed at the top level of input schedulers $\mathbb{S}^{INP}$ satisfy property (3.1) (with upper bound on the service discrepancy of $\mathbb{S}_q$ in (3.1) denoted by $\overline{E}^{IQ}$), then for arbitrary rational speedup $S > 2$, with no assumption on synchronization between the cell slot clock and the phase clock (no alignment of phases in cell slots) the arbitration delay of any cell is upper bounded by*

$$D^A = \frac{\overline{E}^{IQ}(n-1)+1}{S-2} \tag{3.4}$$

where n is the number of input channels in the switch.

Note that since the schedulers $\mathbb{S}_q$ operate on the virtual output queues, the bounds (3.1) and (3.2) should be viewed as the difference in service received by the *aggregate* of all flows multiplexed to this virtual output queue compared to the ideal service the queue should have received if it were scheduled ideally at the aggregate rate assigned to this virtual output queue.

Proof.

Suppose this is not the case. Then there must exist a cell which was scheduled by its scheduler $\mathbb{S}^{INP}$ at its input at some time ξ and was not scheduled by the arbiter at or before time $\xi + D^A$. Let t be the smallest such ξ at which any such cell was scheduled by any of the input schedulers $\mathbb{S}^{INP}$, and denote such earliest "violating" cell by c. Let i and j be c's input and output respectively. Since the arbiter chooses cells on the "smallest timestamp first" basis, and since the timestamp of a cell is its scheduling time under the $\mathbb{S}^{INP}$ scheduler, no cell which was scheduled by its $\mathbb{S}^{INP}$ scheduler after time t could prevent c from being chosen by the arbiter. Further, by the choice of c, no cell scheduled by its $\mathbb{S}^{INP}$ scheduler prior to time $t - D^A$ can remain at its input at time t. Therefore, the only "competition" which could prevent c from being chosen by the arbiter up to and including time $t + D^A$ are those cells that were scheduled by $\mathbb{S}^{INP}$ at c's input in the interval $[t - D^A, t]$, and those cells destined to output j that were scheduled by $\mathbb{S}^{INP}$at all other inputs in this interval. Since D^A is measured in units of channel cell slots, and t is a cell slot boundary, there are at most $D^A + 1$ cell slot boundaries in the interval $[t - D^A, t]$. Since schedulers $\mathbb{S}^{INP}$ make their scheduling decisions at cell slot boundaries only, there are at most $D^A + 1$ cells scheduled by $\mathbb{S}^{INP}$ in the interval $[t - D^A, t]$, including c itself. Note now that the only times when a cell destined to output j from some input $i^{'}$can be scheduled by its scheduler $\mathbb{S}^{INP}$ must correspond to the times when the top-level scheduler $\mathbb{S}_q(i^{'})$ at its input chooses the virtual queue $Q_{i'j}$. Since the $\mathbb{S}_q(i^{'})$ are assumed to satisfy the property (3.1), and since the scheduling decisions at all inputs are made only at discrete time boundaries, it must be that the virtual queue $Q_{i'j}$ was scheduled by $\mathbb{S}_q(i^{'})$ at most $(D^A + \varepsilon)R_{ij} + \overline{E}^{IQ}$ times, (and hence at most $(D^A + \varepsilon)R_{ij} + \overline{E}^{IQ}$ cells destined to output j were scheduled by $\mathbb{S}^{INP}$) at any input i' in the interval $[t-D^A, t] \subset [t-D^A, t+\varepsilon)$ for any arbitrarily small value of ε. Therefore, the maximum number of cells "competing" with c cannot exceed $D^A + (D^A + \varepsilon)\sum_{i'} R_{i'j} + \overline{E}^{IQ}(n-1) \leq 2D^A + (n-1)\overline{E}^{IQ} + \varepsilon$. By Lemma 4, there are at least $D^A S - 1$ arbitration opportunities in the interval $[t, t+D^A]$. It is easy to see that for $S > 2$, choosing sufficiently small ε, for any $D^A > \frac{(n-1)\overline{E}^{IQ}+1}{S-2}$ it must be that $D^A S - 1 > 2D^A + (n-1)\overline{E}^{IQ} + \varepsilon$, and so the number of matching phases exceeds the number of competing cells. Therefore, c could not remain at the input at $t + D^A$. The obtained contradiction proves the Theorem. ■

Note that the arbitration delay is straightforwardly affected by the accuracy of the input scheduler.

3.3.3 A Bound on the Output Delay (The Case of a Cell-switching Network)

This Section considers the output delay of a switch in a cell switching network, where the output scheduler operates directly on cells rather than on re-assembled packets. The discussion of the output delay in a packet-switching network is delayed till Section 3.3.5.

As follows from Theorem 7, the output delay is defined by the value $\underline{E}^{Out}$ of the scheduler employed at the output and the leaky bucket parameters of a flow at the entry to the output channel. While the former is entirely determined by the choice of the output scheduler, as will now be shown, the latter is a function of the arbitration delay (and hence the arbitration algorithm) and the accuracy of the input rate-controlled scheduler. More specifically, the following Theorem holds:

Theorem 9 *If the input rate-controller satisfies (3.1) with work discrepancy bound* $\overline{E}^{INP}$, *and if the arbitration delay is bounded by some time-independent* D^A, *then the flow which conforms to a leaky bucket with parameters* (r, b) *at the entry to the switch conforms to a leaky bucket with parameters* $(r, D^A + \overline{E}^{INP})$ *at its entry to the output channel.*

Proof of Theorem 9.

Consider an arbitrary interval $[t_1, t_2]$. Since it takes exactly time $\frac{1}{S}$ (in channel cell times) to move a cell from the input to the output channel in a switch with speedup S, any cell entering the output at some time $\tau \in [t_1, t_2]$ was scheduled by the arbiter at time $\tau - \frac{1}{S} \in [t_1 - \frac{1}{S}, t_2 - \frac{1}{S}]$. Since this cell could have spent at most time D^A after it passed its input rate-controller, it could pass the input rate-controller anywhere in the interval $[\tau - \frac{1}{S} - D^A, \tau]$. Therefore, a cell arriving to the output channel in the interval $[t_1, t_2]$ could have passed the input rate-controller anywhere in the interval $[t_1 - \frac{1}{S} - D^A, t_2 - \frac{1}{S}]$. By Theorem 11 there could be at most $(t_2 - t_1 + D^A)r + \overline{E}^{INP} \le (t_2 - t_1)r + D^A + \overline{E}^{INP}$ such cells (the inequality following from the fact that the sum of all rates assigned to all flows does not exceed the capacity of an input or output channel, which is assumed to be unit, implying $r \le 1$). Therefore, by the definition of the leaky bucket the traffic of any flow conforms to a leaky bucket with parameters $(r, D^A + \overline{E}^{INP})$ as it enters the output channel. ■

Theorems 7 and 9 now immediately yield

Theorem 10 *The delay at the output channel of any cell of a flow with assigned rate r satisfies*

$$D^{Out} \le \frac{D^A + \overline{E}^{INP} + \underline{E}^{Out}}{r} \tag{3.5}$$

where D^A is the bound on arbitration delay, $\overline{E}^{INP}$ is the work discrepancy bound of the input rate-controlled scheduler (3.1), and $\underline{E}^{Out}$ is the work discrepancy bound of the output scheduler in (3.2).

Finally, the following simple fact characterizes the burstiness of a flow on the output of a generic rate-controller. This fact follows straightforwardly from (3.1):

Theorem 11 *If the scheduler satisfies (3.1) then the flow is constrained by a leaky bucket with parameters $(r, \overline{E})$ at the output from the scheduler.*

Proof.

Consider any interval $[t_1, t_2]$ at the output from the scheduler. The departure of cells from the server is offset by a fixed time interval from the scheduling time (where the offset is simply the time required to transmit one cell at the speed of the server). By (3.1), at most $r(t_2 - t_1) + \overline{E}$ cells could have been served in this interval, which immediately implies the statement of the Theorem. ■

3.3.4 Bounds on the Aggregate Delay and Outgoing Traffic Burstiness

The results of the previous three Sections will now be used to derive the bound on the total delay through the switch of any flow whose burstiness is constrained at the input to the switch. This bound is independent of the number of or the behavior of any other flows.

Theorem 12 *For any speedup $S > 2$ the total switching delay of a any cell of a flow constrained by a leaky bucket (r, b) at the entry to the switch satisfies*

$$D \leq \frac{b + \underline{E}^{INP} + \overline{E}^{INP} + \underline{E}^{Out}}{r} + (1 + \frac{1}{r})\frac{\overline{E}^{IQ}(n-1) + 1}{S - 2} + \frac{1}{S} + \frac{1}{C^{out}} \qquad (3.6)$$

where $\underline{E}^{INP}$ and $\overline{E}^{INP}$ are work discrepancy bounds in (3.1) and (3.2) of the input rate-controlled scheduler $\mathbb{S}^{INP}$, $\overline{E}^{IQ}$ is the upper work discrepancy bound in (3.1) of the top-level scheduler $\mathbb{S}_q$ at the input, $\underline{E}^{Out}$ is the lower work discrepancy bound in (3.2) of the output scheduler, n is the number of input channels in the switch, and C^{out} is the capacity of the output link of this flow.

Proof of Theorem 12.

The aggregate delay at the switch is simply the sum of the input scheduling delay D^{INP}, the arbitration delay D^A, the output scheduling delay D^{Out}, plus the time required to transmit one cell across the switch at the speed of the switch fabric, and also plus the time required to transmit one cell at the speed of the outgoing link. Assuming that the propagation delay across the switch is negligible, the time to transmit the cell across the switch is simply $\frac{1}{S}$. The time to transmit a cell over the output link is $\frac{1}{C^{Out}}$.

Combining the bounds (3.3) and (3.5) on D^{INP} and D^{Out} yields

$$D \leq \frac{b + \underline{E}^{INP} + \overline{E}^{INP} + \underline{E}^{Out}}{r} + (1 + \frac{1}{r})D^A + \frac{1}{S} + \frac{1}{C^{out}} \tag{3.7}$$

Using (3.4) immediately implies the statement of the Theorem. ■

It is easy to see that Theorem 12 quantifies the intuition that the more accurate the schedulers at the input and output channels, the smaller the delay bound.

Therefore, employing accurate schedulers at the input and the output appears essential. The next Chapter discusses some known schedulers and their accuracy.

Finally, Theorem 11 immediately implies that the traffic of a flow at the output from the switch satisfies the following Theorem:

Theorem 13 *The traffic at the output of the switch conforms to a leaky bucket* $(r, \overline{E}^{Out})$

Note that the burstiness of the traffic at the output of the switch does not depend on anything but the upper bound on service discrepancy of the output scheduler. This property follows from the assumption that traffic is reshaped at the output, so the more reshaping is desired, the more accurate traffic shaper should be employed at the output.

3.3.5 Switching Delay in a Packet Network.

The previous Section was dedicated to the derivation of a switching delay bound in a cell-switching network such as ATM. In the case of a packet switching network, however, it is typically the *packet* delay that is of interest.

Recall that in a packet-switching network as soon as a packet arrives to the input channel it is fragmented into cells. When the cells reach the output channel, they are reassembled into packets and only then are passed through the output scheduler.

This Section examines the components of the total switching delay for a packet. It is assumed that an upper bound $L_{\max}$ on the length of a packet is known, and that the time required to fragment a packet is bounded by T^{FRAG} . Similarly, it is assumed that the reassembly delay at the output (once all the cells of a packet have already arrived there) is also bounded by T^{RSMB}.

When a packet is fragmented into cells, typically some header information needs to be added to each cell to allow correct reassembly at the output. Furthermore, if there is not enough data in the packet to fill an integer number of cells, the last cell may be "padded" by some dummy bits. All these issues introduce additional bandwidth overhead, which affects both the effective "cell bandwidth" and "cell burst size" of a flow.

The amount of overhead strongly depends on the particular algorithms used for fragmentation and reassembly (e.g. whether it is allowed to put data from the end of one packet and the beginning of the next packet in one cell). Furthermore, the overhead depends on the distribution of the packet size. This dissertation does not consider specific fragmentation and reassembly algorithms, and therefore this overhead is not quantified. Instead it is assumed that the assigned rate as well as the burst size in the input leaky bucket parameters are already adjusted to accommodate this overhead. Namely, if a flow conforms to a "packet" leaky bucket with parameters (r^p, b^p) at the input, these parameters are adjusted to (r, b) where $r \geq r^p$, and typically $b \geq b^p$. It is assumed that the negotiation of "packet rates" r^p at connection setup takes into account the fragmentation overhead, so that the "cell rates" r_f of all flows sharing an *input* channel still satisfy $\sum_f r_f \leq 1$, while the "packet rates" r_f^p of all flows sharing an output link of capacity C^{Out} satisfy $\sum_f r_f^p \leq C^{Out} \leq 1$.

With this in mind, the rest of this Section will operate with the already adjusted "cell" leaky bucket parameters (r, b).

Consider now the first and the last cell of a packet, which are denoted by c_f and c_l. Since these cells are part of the same packet, their arrival to the switch occurred at the same time (it is assumed that the packet is fully arrived when its last bit has arrived, and that all cells of a packet "arrive" to the switch simultaneously at the end of the fragmentation process). Therefore, the packet scheduling delay is simply the scheduling delay of its *last* cell, and therefore, Theorem 7 implies that for the packet case D^{INP} satisfies (3.3).

Since the packet is considered arrived at the output when its last bit arrives at the output, it is easy to see that the packet arrives to the output channel no later than $D^{FRAG} + D^{INP} + D^{A} + \frac{1}{S}$, which is the bound on the total input scheduling and the arbitration delay of the last cell plus the time required to transmit the last cell at the speed of the switch. Note that input scheduling delay, and the arbitration delay of all the previous cells of the packet are "hidden" in the bound $D^{INP}+ D^{A}$ since by the time the last cell of a packet is chosen by the arbiter, all the previous cells must have been transmitted to the output.

Finally, the packet delay at the output is now evaluated.

Note that Theorem 9 regarding the maximum burst of the leaky bucket parameters of a flow at the entry to the output channel continues to hold, since the size of the maximum burst after packets have been reassembled cannot exceed the size of the maximum burst in cells (before the cell headers is stripped off and possible padding removed).

It is assumed that just as in the case of a cell scheduler at the output, the output *packet* scheduler satisfies (3.1) and (3.2), where the bounds $\overline{E}^{Out}$ and $\underline{E}^{Out}$ may depend on the bounds on the size of the packet[4]. Given this assumption, the proof of Theorem 10 holds without modification (except replacing the word "cell" by "packet"), and so the output scheduling delay is bounded by $D^{RSMB} + D^{Out}$, with D^{Out} satisfying (3.5). Finally, it takes at most $\frac{L_{\max}}{C^{Out}}$ time units to transmit a packet over the output link.

Combining the bounds for D^{INP}, D^{A} and D^{Out} immediately implies that the bound D^{p} total delay of packet in the switch is given

$$D^{p} \leq D^{FRAG} + D^{RSMB} + \tag{3.8}$$

[4]An example of a packet scheduler satisfying (3.1) and (3.2), with tight bounds $\overline{E}^{Out}$ and $\underline{E}^{Out}$ will be given in the next chapter.

$$\frac{b + \underline{E}^{INP} + \overline{E}^{INP} + \underline{E}^{Out}}{r} + (1 + \frac{1}{r})\frac{\overline{E}^{IQ}(n-1)+1}{S-2} + \frac{1}{S} + \frac{L_{\max}}{C^{out}}$$

3.4 Delay Bounds for Small Speedup Values.

So far the delay bounds have been obtained for a switch with speedup $S > 2$. Since the higher the speedup, the more expensive the switch, this Section investigates the possibility of providing bandwidth and delay guarantees with small speedup values. In particular, the values of speedup $1 \leq S \leq 2$ are considered here.

It turns out that even for small speedup values one can still ensure certain delay bounds as long as the total rates of all flows requiring delay guarantees sharing an input or an output channel are restricted to a certain fraction of the channel bandwidth. Namely, it is shown that the total switching delay of any cell is bounded as long as for any input channel i and output channel j the rates of guaranteed flows are restricted to satisfy the following conditions:

$$\begin{aligned} \sum_i R_{ij} &< \alpha, \\ \sum_j R_{ij} &< \alpha \end{aligned} \tag{3.9}$$

where $0 < \alpha < \frac{S}{2}$ and R_{ij} is the combined rate of all flows destined from input i to output j.

Restricting the rates of guaranteed flows to a portion of the channel bandwidth clearly leads to bandwidth under-utilization in the absence of any other classes of service. However, if best-effort traffic is used along with the guaranteed traffic, then the bandwidth unused by the guaranteed flows can be used by best-effort traffic.

With this in mind, the delay bounds for the smaller values of speedup will now be derived.

First, note that the input and output delays D^{INP} and D^{Out} are not affected by the value of the speedup. Hence, it is only needed to obtain the arbitration delay bound, which is given by the following Theorem:

Theorem 14 *If the rate-controllers $\mathbb{S}_q$ at the top level of the schedulers $\mathbb{S}^{INP}$ satisfy property (3.1), and if the rates of all flows are restricted as to satisfy (3.9) with $\alpha < \frac{S}{2}$, then for arbitrary speedup in the range $1 \leq S \leq 2$, with no assumption on synchronization between the cell slot clock and the phase clock, the arbitration delay D^A of any cell is bounded from above by $\frac{\overline{E}^{IQ}(m+n-1)}{S-2\alpha}$*

Proof.

Suppose that the statement of the Theorem is false, and choose a cell c with the earliest scheduling time t under its S_q scheduler, for which the arbitration delay bound is violated. Let i and j be c's input and output channels respectively. Since there are at least $D^A S - 1$ matching opportunities in the interval $[t, t + D^A]$, in order for c to not be scheduled by and including time $t + D^A$ it must be that a cell from input i and/or a cell destined to output j with timestamps smaller than or equal to t (which is the timestamp of c) must be chosen by the arbiter in each of these matching opportunities. Hence, there must be at least $D^A S - 1$ "competing" cells available in the interval $[t, t + D^A]$. Note as in the proof of Theorem 14 that any competing cell must be scheduled by its S_q scheduler in the interval $[t - D^A, t]$, since any cell scheduled before $t - D^A$ must be gone by t by the assumption that c is the first cell to violate the arbitration delay bound D^A, and any cell scheduled by $\mathbb{S}^{INP}$ after time t cannot prevent c from being chosen by the arbiter due to the "smallest timestamp first" arbitration policy. Since the time when a cell is scheduled by $\mathbb{S}^{INP}$ must correspond to the time when the corresponding virtual output queue is chosen by $\mathbb{S}_q$, and since all the schedulers $\mathbb{S}_q$ satisfy (3.1), it means that at most $(\varepsilon + D^A)\sum_j R_{ij} + m\overline{E} < (\varepsilon + D^A)\alpha + m\overline{E}$ cells can be scheduled by the input scheduler $\mathbb{S}^{INP}$ at input i, including c itself, and at most $(\varepsilon + D^A)\sum_{i'} R_{ij} + (n-1)\overline{E} < (\varepsilon + D^A)\alpha + (n-1)\overline{E}$ cells destined to j can be scheduled at all other inputs in the interval $[t - D^A, t] \subset [t - D^A, t + \varepsilon)$ for any arbitrary small values of ε. Therefore, the total number of competing cells is bounded by $2(D^A + \varepsilon)\alpha + (m + n - 1)\overline{E} - 1$.

In order for c not to be scheduled by the arbiter at or before D^A, it must be that $2D^A\alpha + (m+n-1)\overline{E} - 1 + 2\varepsilon\alpha > D^A S - 1$. It is easy to see that for any $\alpha < \frac{S}{2}$, choosing sufficiently small ε, this cannot be true for any $D^A > \frac{\overline{E}(m+n-1)}{S-2\alpha}$ this cannot be true for any $\alpha < \frac{S}{2}$. The obtained contradiction proves the statement of the Theorem. ■

The aggregate switching delay bound can now be immediately obtained from inequality (3.7) using the value $\frac{\overline{E}(m+n-1)}{S-2\alpha}$ for a bound on D^A.

Chapter 4

Choosing Input and Output Schedulers

4.1 Why Use A Rate-Controller at the Input?

In general a work-conserving scheduler which is never idle as long as it has work to do has a lot of intuitive appeal since it eliminates unnecessary loss of bandwidth. However, it turns out that *in the context of the particular timestamp-based arbitration mechanism* described in the previous Chapter a rate controller at the input yields substantially smaller delay bounds compared to those which can be obtained if a work-conserving scheduler is used. Intuitively, this follows from the fact that for a rate-controller, the upper bound on service discrepancy (which in turn defines the arbitration delay bound) does not depend on the shape of the incoming traffic, while as will be shown below, for a work-conserving scheduler this bound is at least linear with the *combined burstiness of all other traffic destined to the same output*. In addition to yielding a larger delay bound, this is quite undesirable since it jeopardizes the principle of flow isolation: a flow with a small burstiness may be affected by a large burst of some other (potentially misbehaved) flow at some other input. In contrast, if a rate-controller is used at the input, the delay bound of a flow does not depend on the burstiness of any other flows.

To get more intuition on this, consider the following example. Consider an nxn switch in a cell-switching network with a work-conserving scheduler at each input, and assume that the arbitration mechanism is timestamp-based as described in Section 3.2.4. Suppose further that there are $n+1$ flows labelled $1, 2...n+1$ sharing a single output. Flows 1 and 2 are from input 1, each with assigned rate $1/2n$, while flows $3, 4...n, n+1$ are from inputs $2, 3, ...n$, each with assigned rate $1/n$. Note that this rate assignment is feasible, since no input or output (each of unit capacity) is overbooked. Suppose further that at the input to the switch all flows $2, 3...n+1$ conform to leaky buckets (r, b) with the same burstiness b, while flow 1 has burstiness b_1. Suppose at time zero b cells of each of flows $2, 3, ...n+1$ arrive to their corresponding inputs. They are scheduled by the input schedulers in a FIFO order, so that the b-th cell is scheduled at time b. Since all cells in the switch share a single output, only a single cell can be chosen by the arbiter at each matching opportunity. Therefore, in b cell times exactly bS cells can be sent across the switch, and at time b $nb - bS$ cells will still remain at their inputs. All of these cells will have timestamps not exceeding b, since they were scheduled by the input schedulers by time b. Suppose now that at time $b+1$ a single cell of flow 1 arrives at input 1. It will be immediately scheduled by the input scheduler and will be assigned timestamp $b+1$. Since it will have the largest timestamp among all cells across the switch, all sharing the same output, it will have to wait until all the other cells are transmitted to the output, yielding the total input plus arbitration delay of $\frac{nb-bS}{S} = \frac{nb}{S} - b$. As can be seen, this delay is proportional to *n times the burst size of the other flows.* In contrast, if a two-level rate-controlled scheduler with constant work discrepancy bounds were used at the input, the total input plus arbitration delay would be given by $\frac{b_1+\underline{E}^{INP}}{r} + \frac{\overline{E}^{IQ}(n-1)+1}{S-2}$, which does not depend on the combined burstiness nb of other flows at all. This example illustrates that for the switch of the same size and speedup, in the presence of some flows with large burstiness b, using a rate-controlled scheduler with constant service discrepancy bounds is beneficial compared to using a work-conserving scheduler.

An example of a rate-controlled scheduler with a small bound on service discrepancy is discussed in the rest of this Chapter.

Note finally that the considerations discussed in this Section have no effect on whether a rate-controller or a work-conserving scheduler is chosen at the *output* channel. There appears to be no overwhelming advantage of one approach versus the other.

4.2 Operation of RC-WF^{2}Q.

The rate-controllers at the input and output channels, which will be used in this Chapter, are based on the so-called RC-WF^{2}Q [4], which is a rate-controlled version of WF^{2}Q [2]. The choice of RC-WF^{2}Q is motivated by its high accuracy, yielding small service discrepancy bounds. The next Section is dedicated to the description of RC-WF^{2}Q and its basic properties.

4.2.1 The Cell RC-WF^{2}Q.

The description of RC-WF^{2}Q presented here is equivalent to that of [4]. The scheduler maintains two state variables per flow: s_f and f_f. These variables have the meaning of the ideal starting and finishing transmission time of the HOL cell of q_f, where the ideal time is computed in reference to the flow's "fluid" model, in which cells of this flow are infinitely divisible and the flow is transmitted continuously at the constant rate r_f. Initially $s_f = 0$, $f_f = \frac{1cell}{r_f}$ for all flows f, where the rates of the flows are measured in cells per unit time. In addition, RC-WF^{2}Q maintains a single variable *now*, which is simply equal to the current real time. If the server capacity is C cells/unit time, then *now* increases by $\frac{1\ cell}{C}$ each scheduling opportunity, because it takes exactly $\frac{1\ cell}{C}$ units of time to transmit a cell at speed C. All flows satisfying the condition $s_f \leq now$ are referred to as *eligible* flows at time *now*. At each cell slot boundary RC-WF^{2}Q picks the flow with the smallest finish time f_f among all eligible flows. If a flow f is chosen at the current cell boundary, its state variables are updated as

$$s_f \Leftarrow s_f + \frac{1}{r_f} \tag{4.1}$$

$$f_f \Leftarrow f_f + \frac{1}{r_f}. \tag{4.2}$$

If a flow is not chosen, its state variables remain unchanged. In contrast with a non-rate-controlled version of WF^{2}Q, the flow is scheduled regardless of whether it actually has a cell or not. If a flow is scheduled when there is no cell in it, the flow simply misses the scheduling opportunity (but note that the state variables of the flow are updated as if the cell were there).

The following Theorem and Corollaries characterize the accuracy of RC-WF^{2}Q:

Theorem 15 *For all $k \geq 1$, the k-th scheduling opportunity under RC-WF^{2}Q of a flow with assigned rate r occurs in the interval $[\frac{k-1}{r}, \frac{k}{r}]$.*

This Theorem can be derived from the results in [2], [4]. Below is an alternative proof derived from the properties of PGPS [20].

Proof of Theorem 15.

Consider first the case when the system is fully booked, i.e. $\sum_f r_f = 1$. The operation of RC-WF^{2}Q is independent of any arrival pattern, since the scheduling opportunities occur *as if* for all $k \geq 1$ the k-th cell would arrive exactly at its ideal arrival time $\frac{k-1}{r}$. Therefore, for the fully booked case the operation of RC-WF^{2}Q is equivalent to the operation of PGPS under the ideal arrivals of all cells of all flows. Theorem 1.1 of [20] implies that in a fully booked PGPS scheduler operating at unit speed on unit size packets (cells) arriving exactly at their ideal arrival time, the k-th cell finishes its transmission no later than at time $\frac{k}{r} + 1$. Since it takes exactly unit time to transmit a cell at unit speed, it means that the k-th scheduling opportunity (i.e. the time at which the k-th cell *begins* its transmission) occurs no later than at time $\frac{k}{r}$, where r is the rate assigned to this flow. Moreover, since under the considered ideal arrival pattern the k-th cell arrives exactly at time $\frac{k-1}{r}$, it clearly cannot be transmitted before that time. Hence, under PGPS with ideal arrivals in the fully booked case, the k-th scheduling opportunity of any flow occurs in the interval $[\frac{k-1}{r}, \frac{k}{r}]$, immediately yielding the statement of the Theorem for the fully booked RC-WF^{2}Q scheduler.

It remains to show that the Theorem is true for the case when $\sum_f r_f < 1$. Consider a dummy flow (which will be denoted by $\widetilde{f}$ with rate $\widetilde{r} = 1 - \sum_f r_f$. Assume that the "dummy" cells of this "dummy" flow arrive at their ideal arrival times. Let $\tau_f(k)$ and $t_f(k)$ denote the time of the k-th scheduling opportunity of some flow f under RC-WF^{2}Q in the presence of the dummy flow and without it, respectively. Note that for all k the *values* of variables s_f and f_f assigned at the k-th update of these variables do not depend on the *time* of the update or the state variables of any other flow in the system. Therefore, although the presence of the dummy flow may change the *scheduling times* of the real flows, it cannot change the *relative scheduling order* of the real flows. This implies that the presence of the dummy flow can only cause the k-th scheduling opportunity of a real flow to occur later than that in the absence of the dummy flow. Hence, for all k $t_f(k) \leq \tau_f(k)$. As shown above, for the fully booked case $\tau_f(k) \in [\frac{k-1}{r}, \frac{k}{r}]$, and hence $t_f(k) \leq \tau_f(k) \leq \frac{k}{r}$. On the other hand, by operation of the algorithm no cell can be scheduled before its eligibility time, implying $t_f(k) \geq \frac{k-1}{r}$. Hence, $t_f(k) \in [\frac{k-1}{r}, \frac{k}{r}]$. ■

Corollary 16 *Service discrepancy bounds of an RC-WF^{2}Q scheduler satisfy* $\underline{E}^{RCWF^2Q} = 2$ *cells and* $\overline{E}^{RCWF^2Q} = 2$ *cells.*

Proof of Corollary 16.

It will be shown below that the number of scheduling opportunities (measured in cells) of a flow with assigned rates r which can occur in any interval $[t_1, t_2)$, which we denote by $A_r(t_1, t_2)$, satisfies

$$(t_2 - t_1)r - 2 < A_r(t_1, t_2) < (t_2 - t_1)r + 2 \tag{4.3}$$

Since $(t_2 - t_1)r$ is the amount of "ideal" service which should be received in fluid at rate

r, the statement of the Corollary will follow.

Let $\frac{m-1}{r} \leq t_1 < \frac{m}{r}$, $\frac{m+p-1}{r} \leq t_2 < \frac{m+p}{r}$, for some $m \geq 0$, $p \geq 1$. Clearly,

$$\frac{p-1}{r} = \frac{m+p-1}{r} - \frac{m}{r} < t_2 - t_1 < \frac{m+p}{r} - \frac{m-1}{r} = \frac{p+1}{r} \tag{4.4}$$

By Theorem 15 all scheduling opportunities of the considered flow indexed $m+1$,$m+2, ...m+p-1$ must occur in the interval $[t_1, t_2]$. Since there are at least $p-1$ of them, it must be that $A_r(t_1, t_2) \geq p-1$. In conjunction with (4.4) this implies that

$$A_r(t_1, t_2) \geq p-1 > (t_2 - t_1)r - 2$$

which is the lefthand side of (4.3). On the other hand, by Theorem 15 only scheduling opportunities of the considered flow numbered $m, m+1, ...m+p$ can possibly occur in the interval $[t_1, t_2)$. Since there are at most $p+1$ of those, it must be that $A_r(t_1, t_2) \leq p+1$. In conjunction with (4.4) this implies that

$$A_r(t_1, t_2) \leq p+1 < (t_2 - t_1)r + 2$$

which is the righthand side of (4.3). ■

4.2.2 The Input scheduler based on RC-WF^{2}Q.

The input rate-controlled scheduler was described in the previous Chapter as a two-level hierarchical scheduler in which the top level scheduler controls the input to the virtual output queues, and the second level schedulers operate on per-flow queues. In this context, an RC-WF^{2}Q scheduler is used at the top level of the hierarchy to arbitrate among virtual output queues at each input (that is, all $\mathbb{S}_q(i)$ schedulers are RC-WF^{2}Q schedulers as described in the previous Section).

At the second level of the hierarchy at input i there are n RC-WF^{2}Q schedulers $\mathbb{S}_f(i.j)$, one for each of the n output channels. Unlike the top-level scheduler, $\mathbb{S}_f(i,j)$

does not operate in real time, but rather is invoked any time the top-level RC-WF^{2}Q chooses the virtual output queue Q_{ij}. The only distinction between the operation of the second-level schedulers $\mathbb{S}_f(i,j)$ and the non-hierarchical RC-WF^{2}Q described above is in the meaning and the value of the variable *now*, which we will denote τ to distinguish from the variable *now* of the top level of the hierarchy . While *now* is simply the real time in the top-level scheduler, in $\mathbb{S}_f(i,j)$ τ advances by $\frac{1}{R_{ij}}$ only at the times when Q_{ij} is chosen by the top-level scheduler, after the scheduling decision of $\mathbb{S}_f(i,j)$ is made. More specifically, initially $now = \tau = 0$. If at real time $now = t$ the top-level RC-WF^{2}Q chooses Q_{ij}, then $\mathbb{S}_f(i,j)$ chooses the flow with the smallest f_f of all of its flows with $s_f \geq \tau$, and then updates $\tau \leftarrow \tau + \frac{1}{R_{ij}}$. The variable τ will be referred to as "simulated time" to distinguish it from the real time.

Theorem 17 *For all $k \geq 1$ the k-the scheduling opportunity of flow f with assigned rate r_f under the two-level hierarchical RC-WF^{2}Q occurs in the interval $[\frac{k-1}{r_f}, \frac{k+1}{r_f}]$*

This Theorem can be derived from the results of [3] for hierarchical WFQ schedulers. A more direct proof is given below.

Proof of Theorem 17.

Consider a scheduler $\mathbb{S}_f(i,j)$ in the hierarchy and let $t_1, t_2, ...$ be the sequence of real times corresponding to the time epochs when variables s_f and f_f changed for at least one flow in the scheduler. Let $\tau_1, \tau_2, ...$ be the corresponding values of the simulated time variable τ of this scheduler. By operation of the algorithm $\tau_1 = 0, \tau_2 = \frac{1}{R_{ij}}, ...\tau_k = \frac{k-1}{R_{ij}}$. Consider now an isolated WF^{2}Q scheduler operating on the same flows with the same rate assignments on a link of capacity R_{ij}. Let $\widetilde{t}_1 = 0, \widetilde{t}_2 = \frac{1}{R_{ij}}, ...\widetilde{t}_k = \frac{k-1}{R_{ij}}, ...$ be the real times corresponding to "cell times" at this link speed. It is easy to see that although the real times $t_1, t_2, ...$ may differ from $\widetilde{t}_1, \widetilde{t}_2.....$,the sequence of simulated times $\tau_1, \tau_2, ...$ is identical to $\widetilde{t}_1, \widetilde{t}_2.......$ Furthermore, by operation of the algorithm the values of variables s_f and f_f at times $t_1, t_2, ...$ for the scheduler in the hierarchy are identical to the corresponding values of these variables at times $\widetilde{t}_1, \widetilde{t}_2......$ for the isolated WF^{2}Q scheduler.

Consider now the *real time* t_k of the k-th scheduling opportunity of flow f with the $\mathbb{S}_f(i,j)$ scheduler. This time can occur only at a time when the corresponding Q_{ij} is scheduled under the top-level RC-WF^{2}Q. Let this be the m-th scheduling opportunity of Q_{ij} under the top-level RC-WF^{2}Q. By Theorem 15 applied to the top level RC-WF^{2}Q,

$$\frac{m-1}{R_{ij}} \leq t_k \leq \frac{m}{R_{ij}} \tag{4.5}$$

By operation of the algorithm, the variable τ of $\mathbb{S}_f(i,j)$ at time t_k (at the time the scheduling decision is made, before the update of τ) which we denote as $\tau(t_k)$ is given by

$$\tau(t_k) = \frac{m-1}{R_{ij}} \tag{4.6}$$

Applying Theorem 15 to the $\mathbb{S}_f(i,j)$ operating in isolation on a link of capacity R_{ij} we get

$$\frac{k-1}{r_f} \leq \tau(t_k) \leq \frac{k}{r_f} \tag{4.7}$$

(4.6) and (4.7) imply

$$\frac{k-1}{r_f}R_{ij} + 1 \leq m \leq \frac{k}{r_f}R_{ij} + 1$$

which, together with (4.5) yields

$$\frac{k-1}{r_f} \leq \frac{m-1}{R_{ij}} \leq t_k \leq \frac{m}{R_{ij}} \leq \frac{k}{r_f} + \frac{1}{R_{ij}} \leq \frac{k+1}{r_f}$$

where the last inequality follows from the fact that $R_{ij} \geq r_f$, since R_{ij} is the sum of rates of all flows destined from i to j. This completes the proof of the Theorem. ■

Corollary 18 *Service discrepancy bounds of a two-level hierarchical RC-WF^{2}Q satisfy* $\underline{E} = 3$ *cells and* $\overline{E} = 3$ *cells.*

Proof of Corollary 18.

The proof is very similar to that of Corollary 16. It will be shown that the number $A_r(t_1, t_2)$ of scheduling opportunities of a flow with assigned rates r which can occur in any interval $[t_1, t_2]$ satisfies

$$(t_2 - t_1)r - 3 < A_r(t_1, t_2) < (t_2 - t_1)r + 3 \tag{4.8}$$

Since $(t_2 - t_1)r$ is the amount of "ideal" service which should be received in fluid at rate r, the statement of the Corollary will follow.

Let $\frac{m-1}{r} \leq t_1 < \frac{m}{r}$, $\frac{m+p-1}{r} \leq t_2 < \frac{m+p}{r}$, $m \geq 0$, $p \geq 1$. Clearly,

$$\frac{p-1}{r} = \frac{m+p-1}{r} - \frac{m}{r} < t_2 - t_1 < \frac{m+p}{r} - \frac{m-1}{r} = \frac{p+1}{r} \tag{4.9}$$

By Theorem 17 all scheduling opportunities of the considered flow indexed $m+1$,$m+2, ...m+p-2$ of the given flow must occur in the interval $[t_1, t_2]$. Since there are at least $p-2$ such opportunities, it must be that $A_r(t_1, t_2) \geq p-2$. In conjunction with (4.9) this implies that

$$A_r(t_1, t_2) \geq p - 2 > (t_2 - t_1)r - 3$$

which is the lefthand side of (4.8). On the other hand, by Theorem 15 only scheduling opportunities of the considered flow numbered $m-2, m-1, ...m+p-1$ can possibly occur in the interval $[t_1, t_2]$. Since there are at most $p+2$ of those, it must be that $A_r(t_1, t_2) \leq p+1$. In conjunction with (4.9) this implies that

$$A_r(t_1, t_2) \leq p + 2 < (t_2 - t_1)r + 3$$

which is the righthand side of (4.8). ■

4.2.3 Switching Delay Bounds

Consider first the case of a cell-switching network, and assume that the output channel employs a single-level RC-WF^{2}Q scheduler, while the input runs a two-level RC-WF^{2}Q scheduler as discussed in the previous Section. The arbitration algorithm uses scheduling times of the input scheduler as discussed in Section 3.2.4. The results of the previous two Sections yield $\overline{E}^{INP} = \underline{E}^{INP} = 3$, $\underline{E}^{IQ} = \overline{E}^{IQ} = \underline{E}^{Out} = 2$.

The total switching delay bound can now be immediately obtained by substituting these values into the general delay bound of Theorem 12:

Theorem 19 *In a cell-switching network, for any speedup $S > 2$ the total switching delay of a any cell of a flow constrained by a leaky bucket (r, b) at the entry to the switch satisfies*

$$D \leq \frac{b+8}{r} + (1+\frac{1}{r})\frac{2n-1}{S-2} + \frac{1}{S} + \frac{1}{C^{out}}$$

Consider now a packet-switching network. In this case a packet scheduler is used at the output. Since either a rate-controller or a work-conserving scheduler can be used at the output, suppose that a work-conserving packet WF^{2}Q scheduler is used at the output. As follows from the results in [2], its delay satisfies $D^{Out} = D^{WF2Q} \leq \frac{\widetilde{b}+L_{\max}}{r}$, where $L_{\max}$ is the length of the longest packet, and $\widetilde{b}$ is the burstiness of the flow at the input to the scheduler (i.e. at the entry to the output channel). Since the input delay D^{INP} and the arbitration delay D^A are the same as for the cell-switching network, and the leaky-bucket parameters of the flow at the entry to the switch is given by Theorem 9, the total delay in a packet switching network becomes

$$\begin{aligned} D^p \leq\ & D^{FRAG} + D^{RSMB} + \\ & \frac{b+6+2L_{\max}}{r} + (1+\frac{1}{r})\frac{2n-1}{S-2} + \frac{1}{S} + \frac{L_{\max}}{C^{out}} \end{aligned}$$

where as before D^{FRAG} and D^{RSMB} denote the fragmentation and reassembly delay.

Chapter 5

Arbitration Delay Bounds Independent of the Switch Size.

5.1 A Limitation of Arbitration Based on Input Scheduling Times.

The scheduling mechanisms in the previous Chapters are conceptually simple and provide deterministic bandwidth and delay guarantees to leaky-bucket constrained flows independently of the behavior of the other flows. The main weakness of the described approach, however, is that the resulting delay bounds are a function of the size of the switch. While the linear dependence of the arbitration delay on the size of the switch may be acceptable for small switches, it clearly is very unfortunate when the number of input channels becomes large. It is therefore desirable to design algorithms that would provide delay guarantees independent of the size of the switch. This Chapter is dedicated to a description of an algorithm which yields delay bounds independent of the size of the switch.

The arbitration algorithm discussed in the previous Chapters was based on the timestamps corresponding to the scheduling times of the input rate-controller $\mathbb{S}^{INP}$. In this framework, the input scheduler is aware of the rate assignments, while the arbiter is completely oblivious to them, relying on the inputs to schedule different flows at the correct rate. It is this "rate-blindness" of the arbiter that causes the arbitration delay bound to be linear in the number of input channels. This can be trivially seen by considering an example in which a single output is shared by n flows, indexed $1...,n$ each arriving at a different input channel. Assume that the rate r_1 of flow 1 is very close to the (unit) channel capacity, while the rates r_i $(i = 2,..n)$ of all other flows $r_i = \frac{1-r_1}{n-1}$ are very close to zero. Suppose that a cell of each of the flows $1,..n$ arrives at time zero to the corresponding input channels, and are immediately scheduled by the input rate-controllers (in the absence of any other cells at the inputs). Then all of these n cells are assigned the timestamp $t = 0$. Suppose now that the arbiter breaks ties by choosing the input with the largest index first. Then the cell of flow 1 will have to wait $n - 1$ matching phases before it can be transmitted to the output, so its arbitration delay is $\frac{n-1}{S}$ which is linear in n. Note that since the rate of flow 1 is close to 1, $\frac{n-1}{S} \approx \frac{n-1}{S}\frac{1}{r_1}$, i.e. the arbitration delay, as expressed in ideal inter-scheduling intervals of the flow, grows linearly with the size of the switch.

The next section is dedicated to a description of an algorithm which is proven to provide a total switching delay bound which is independent of the size of the switch with speedup $S \geq 6$. In this algorithm the arbiter is made aware of the flow rates.

5.1.1 The Basic Algorithm

We begin by describing an algorithm which is easy to analyze but which suffers from high complexity of the arbiter, since it needs to maintain individual state information for all flows. We then show how to reduce this load by distributing some of the state information and the computational load among the input channels. Consider first the case when the input channels have per-flow queues (but no virtual output queueing). In this framework, the input channel simply stores the arriving cells in the appropriate queues where they wait for the arbiter to decide when to transfer them to the output.

The arbiter maintains the following per-flow information: the rate r_f of each flow f, and the ideal start time of the HOL cell of f in the queue, denoted as before by s_f. As in the rate-controlled version of WF^{2}Q described above, we call flows with $s_f \leq t$ *eligible at time* t. Initially, $s_f = 0$. The arbiter computes the maximal matching as follows:

1. if no eligible flows, stop, else initialize set $\Psi = \{\text{all eligible flows}\}$;

2. pick flow $f \in \Psi$ with the highest rate, breaking ties arbitrarily; put f in the matching[1] and update $s_f \Leftarrow s_f + \frac{1}{r_f}$

3. remove from Ψ all flows with the same input or output as the flow f picked in step 1; if $\Psi = \varnothing$, stop, else go to step 2

This algorithm will be referred to as *Fastest Rate Eligible Cell First* (FRECF). Note that being a rate-controller, FRECF does not concern itself with whether or not the flow it schedules (puts in the match) at any given time actually has a cell or not. This is motivated primarily by the assumption that guaranteed traffic is run at a higher priority, and that any arbitration opportunities missed by a guaranteed flow are used by best-effort traffic.

5.1.2 A Delay Bound for Speedup $S = 6$.

The following Theorem characterizes the accuracy of FRECF:

Theorem 20 *With speedup $S \geq 6$ the k-th scheduling opportunity of flow f with rate r_f under FRECF occurs in the interval $[\frac{k-1}{r_f}, \frac{k}{r_f}]$.*

Proof. Just as in the case of RC-WF^{2}Q, it is convenient to imagine that the scheduler is backlogged with "dummy" cells which are replaced by real cells upon their arrival. We first show that the first scheduling opportunity of any flow f occurs in the interval $[0, \frac{1}{r_f}]$

[1]Note here that in step 2 a flow is chosen (and its start time is updated) even if there is actually no cell in the flow's queue.

for any $S \geq 4$. Note that the first cell of each flow becomes eligible at time 0. Consider any flow f destined from input i to output j and suppose that its first cell c is not scheduled up to and including time $\frac{1}{r_f}$. By the operation of the algorithm this means that at each matching phase, an eligible cell, either from input i or destined to output j, with $r_{f'} \geq r_f$ was scheduled by the arbiter. Note now that for any such flow f', only cells with eligibility times in the interval $[0, \frac{1}{r_f}]$ can prevent c from being scheduled in the interval $[0, \frac{1}{r_f}]$. It is easy to see that for any $r_{f'} \geq r_f$ there could be at most $\frac{r_{f'}}{r_f} + 1 \leq 2\frac{r_{f'}}{r_f}$ such cells. Recalling that the sum of rates of all flows at any input (or destined to any output) does not exceed 1, it follows that there are at most $\sum_{f'} 2\frac{r_{f'}}{r_f} \leq \frac{2}{r_f}$ cells with eligibility time not exceeding $\frac{1}{r_f}$ at input i, which includes c itself. Therefore, the maximum number of c's competitors at the input is at most $\frac{2}{r_f} - 1$, and similarly, there are at most $\frac{2}{r_f} - 1$ competing cells destined to output j. Therefore there are at most $\frac{4}{r_f} - 2$ cells that can prevent c from being scheduled in the interval $[0, \frac{1}{r_f}]$. On the other hand, by Lemma 4 there are at least $\frac{S}{r_f} - 1$ matching phase boundaries in the interval $[0, \frac{1}{r_f}]$, and hence there will not be enough competing cells for any $S \geq 4$.

We now show that for $S \geq 6$ the statement of the Theorem holds for any $k > 1$ as well. Note that by operation of the algorithm a cell cannot be scheduled before its eligibility time. Therefore, we only need to prove that the k-th cell of each flow is scheduled before $\frac{k}{r_f}$, which is its ideal finishing time in fluid. Suppose that the statement of the Theorem is false. This means that there exists some flow f and some $k \geq 2$ such that the k-th cell of f was not scheduled by $\frac{k}{r_f}$. We call such cell a violating cell. Consider the violating cell c with the smallest eligibility time of all violating cells, and let f be the flow it belongs to. Assume that f is destined from an input i to an output j. Let k be c's index, so that its eligibility time is $\frac{k-1}{r_f}$. In order for c to not be transmitted by its deadline $\frac{k}{r_f}$, it must be that at each matching phase boundary in the interval $[\frac{k-1}{r_f}, \frac{k}{r_f}]$ an eligible cell with faster rate was scheduled either from the input i, and/or to the output j. Note now that by the choice of c, it must be that all cells with ideal finish times less than $\frac{k-1}{r_f}$ must have been transmitted by $\frac{k-1}{r_f}$. Therefore, the only "competition" to c that can prevent it from being scheduled by its deadline $\frac{k}{r_f}$ are those cells whose rates are larger, whose eligibility times do not exceed $\frac{k}{r_f}$, and whose finish times are at least $\frac{k-1}{r_f}$. It is easy to see that for any f' with $r_{f'} \geq r_f$ there could be at most $\frac{r_{f'}}{r_f} + 2 \leq 3\frac{r_{f'}}{r_f}$ such cells. Summing as earlier these cells over all flows with rates $r_{f'} \geq r_f$ at c's input and output, we see that the maximum amount of competition that can prevent c from being scheduled by $\frac{k-1}{r_f}$ is at most $\frac{6}{r_f} - 2$, which falls short of the number of matching phase boundaries in the interval $[\frac{k-1}{r_f}, \frac{k}{r_f}]$, which is at least $\frac{6}{r_f} - 1$. The obtained contradiction completes the proof of the Theorem. ■

Corollary 21 *A cell of flow f arriving to the switch at time t and finding Q cells in its queue is delivered to the output no later than at time $t + \frac{Q+2}{r_f} + \frac{1}{S}$ for $S \geq 6$.*

Proof.

Consider a cell c arriving to the switch at some time t. Suppose first that at time t there are no other cells in the flow's queue, i.e. $Q = 0$ and c starts a busy period. Let $\frac{k-1}{r_f} \leq t < \frac{k}{r_f}$ for some $k \geq 1$. Since the k-th scheduling opportunity of this flow might have occurred prior to t in the interval $[\frac{k-1}{r_f}, t)$, the cell c may need to wait till the next scheduling opportunity, which is guaranteed to occur in the interval $[\frac{k}{r_f}, \frac{k+1}{r_f}]$, so c will be scheduled no later than at time $t + \frac{2}{r_f}$, and will be delivered to its output at most by $t + \frac{2}{r_f} + \frac{1}{S}$.

Suppose now that $Q > 0$ at time t, i.e. the cell c arrives to a busy queue. Let $c_1, c_2 ... c_Q$ be the sequence of cells in the queue ahead of c, staring from the head-of-the-line cell. Using the same argument as above, it must be that cell c_1 is scheduled by time $\frac{k+1}{r_f}$, which implies that cell c_2 will be scheduled at the next scheduling opportunity by time $\frac{k+2}{r_f}$, and, inductively, cell c_Q will be scheduled by time $\frac{k+Q}{r_f}$, and finally the cell c itself will be scheduled no later than by time $\frac{k+Q+1}{r_f}$. Recalling that $\frac{k-1}{r_f} \leq t < \frac{k}{r_f}$ it immediately follows that the cell c is scheduled no later than by time $t + \frac{Q+2}{r_f}$, and so it is guaranteed to arrive to its output by time $t + \frac{Q+2}{r_f} + \frac{1}{S}$. ■

Corollary 22 *If the input traffic of flow f with assigned rate r_f conforms to a leaky bucket (r_f, b) then the queue of this flow is bounded by $b + 1$ for $S \geq 6$.*

Proof.

Suppose this is not the case. Then there must exist some time t at which there are at least $b+2$ cells of the considered flow in the queue. Let t_0 denote the beginning of the flow's busy period containing t. Since the flow is constrained by a leaky bucket (r_f, b), at most $(t-t_0)r_f + b + \varepsilon r_f$ cells could have arrived to the queue in the interval $[t_0, t+\varepsilon)$ for any $\varepsilon > 0$. On the other hand, it follows from Theorem 20 that at least $(t-t_0)r_f - 1$ scheduling opportunities of this flow must have occurred in the interval $[t_0, t] \subset [t_0, t+\varepsilon)$. Furthermore, since the queue has been continuously backlogged in the interval $[t_0, t]$, a cell of this flow was actually transmitted at each of these scheduling opportunities. Therefore, for any $\varepsilon > 0$ the queue is bounded from above by $b+1+\varepsilon r_f$. Since ε can be chosen arbitrarily small, the statement of the Corollary follows.

These two last Corollaries immediately imply the following Theorem:

Theorem 23 *For any flow conforming to a leaky bucket (r_f, b) at the input to the switch, the delay of any cell between its arrival to the switch and its delivery to the output channel is bounded by $\frac{b+3}{r_f} + \frac{1}{S}$ for $S \geq 6$.*

Theorem 24 *Flow f conforms to a leaky bucket $(r_f, 2)$ at the entry to the output channel regardless of the shape of its traffic at the input to the switch for $S \geq 6$.*

Proof.

Consider any interval $[t_1, t_2)$ and let

$$\frac{k-1}{r_f} \leq t_1 < \frac{k}{r_f} \tag{5.1}$$

$$\frac{k+m-1}{r_f} \leq t_2 < \frac{k+m}{r_f} \tag{5.2}$$

for some integer $k \geq 1, m \geq 0$. By Theorem 20 at most $m+1$ cells can be scheduled in $[t_1, t_2)$. From (5.1) $k > t_1 r_f$. From (5.2) $m \leq t_2 r_f - k + 1$. Therefore $m+1 \leq t_2 r_f - k + 2 \leq (t_2 - t_1) r_f + 2$. By definition this implies that the flow conforms to a leaky bucket $(r_f, 2)$.■

Finally, using this Theorem in conjunction with Theorems 7 and 23 yields the following Theorem:

Theorem 25 *The total switching delay of any cell of a flow constrained by a leaky bucket* (r_f, b) *at the entry to the switch is* $\frac{b+\underline{E}^{Out}+5}{r_f} + \frac{1}{S}$, *where* $\underline{E}^{Out}$ *is the lower bound on the work discrepancy of the output scheduler.*

5.1.3 A Delay Bound for a Range of Speedup Values - a Generalization.

The previous Section demonstrated that $S = 6$ is sufficient to guarantee that any flow with rate r will be ensured that its k–th scheduling opportunity will occur in the interval $[\frac{k-1}{r}, \frac{k}{r}]$. In this Section it will be shown that if the requirement on the accuracy of the FRECF arbitration is relaxed to require that the k-th scheduling opportunity occurs in the half-interval $[\frac{k-1}{r}, \frac{k+a}{r})$ for some $a \geq 0$, then a lower speedup value (which depends on the desired value of parameter a) will suffice to provide such accuracy.

We now prove the following Theorem:

Theorem 26 *For any* $a \geq 0$, *if* $S > 4 + \frac{2}{a+1}$ *then the k-th scheduling opportunity of flow* f *with assigned rate* r_f *occurs in the interval* $[\frac{k-1}{r_f}, \frac{k+a}{r_f})$.

Proof.

Just as in the case of RC-WF^{2}Q, it is convenient to imagine that the scheduler is backlogged with "dummy" cells which are replaced by real cells upon their arrival. Suppose that the statement of the Theorem is not true. Then there must exist some cell of some flow f such that the statement of the Theorem is violated. Consider the cell c with the earliest eligibility time of all such violating cells of all flows (breaking ties arbitrarily). Let r_f be the rate of the flow f to which c belongs, and let $k \geq 0$ be the sequence number of c, so that $\frac{k-1}{r_f}$ is its eligibility time. In order for c not be scheduled in the interval $[\frac{k-1}{r_f}, \frac{k+a}{r_f})$, it must be that a cell either from c's input or destined to c's output which was also eligible and belonged to some flow ϕ (which may also be f itself) with $r\phi \geq r_f$.

Note now that any such cell competing with c must have become eligible at some time $\tau \geq \frac{k-1-(a+1)}{r_f}$. To see this, note that if a cell $\widetilde{c}$ became eligible at some time $\varsigma = \frac{m-1}{r\phi}$ $< \frac{k-1-(a+1)}{r_f} < \frac{k-1}{r_f}$ for some $m \geq 0$ (and was not yet scheduled by time $\frac{k-1}{r_f}$ to belong to the competition to c), then $\widetilde{c}$ will not have been scheduled in the interval $[\frac{m-1}{r\phi}, \frac{m+a}{r\phi})$ because $\frac{m+a}{r\phi} = \frac{m-1+(a+1)}{r\phi} \leq \varsigma + \frac{a+1}{r_f} < \frac{k-1-(a+1)}{r_f} + \frac{a+1}{r_f} = \frac{k-1}{r_f}$, and hence $\widetilde{c}$ would be a violating cell with an earlier eligibility time than c, which would contradict the choice of c.

As a result, the only competition that could prevent c from being scheduled in the interval $[\frac{k-1}{r_f}, \frac{k+a}{r_f})$ must be those cells that are

1. eligible in the interval $[\frac{k-1-(a+1)}{r_f}, \frac{k+a}{r_f})$

2. share the same input or output with c

3. belong to flows which have rates higher than r_f

Let n_i and n_o denote the number of flows with rates higher than r_f at f's input and output. Since the sum of all rates sharing an input (or an output) is bounded by 1, then for all such flows ϕ at the same input with f $(n_i + 1)r \leq r + \sum_{\phi:r\phi\geq r} r\phi \leq 1$, and hence $n_i \leq \frac{1}{r} - 1$, and analogously $n_o \leq \frac{1}{r} - 1$.

Further, for any flow with rate $r\phi$ there are at most $\frac{2a+2}{r_f}r\phi$ $+1$ cells whose eligibility time is in the interval $[\frac{k-1-(a+1)}{r_f}, \frac{k+a}{r_f})$. Therefore, there are at most $\frac{2a+2}{r_f}\sum_{\phi:r\phi\geq r} r\phi$ $+n_i$ $\leq \frac{2a+2}{r_f} + \frac{1}{r_f} - 1 = \frac{2a+3}{r} - 1$ competing cells at f's input, and at most $\frac{2a+2}{r_f}\sum_{\phi:r\phi\geq r} r\phi$ $+n_o$ $\leq \frac{2a+2}{r_f} + \frac{1}{r_f} - 1 = \frac{2a+3}{r_f} - 1$ at f's output. Therefore, the total amount of competition cannot exceed $\frac{4a+6}{r_f} - 2$. Since there are at least $\frac{a+1}{r_f}S - 1$ arbitration opportunities in the interval $[\frac{k-1-(a+1)}{r_f}, \frac{k+a}{r_f})$, in order for c to be a violating cell it must be that $\frac{a+1}{r_f}S - 1 \leq \frac{4a+6}{r_f} - 2$, or $S \leq (\frac{4a+6}{r_f} - 1)\frac{r_f}{a+1}$. Therefore, as long as $S > 4 + \frac{2}{a+1} - \frac{r_f}{a+1}$, the assumption that the statement of the Theorem is invalid leads to a contradiction. In particular, since $4+\frac{2}{a+1} > 4+\frac{2}{a+1}-\frac{r_f}{a+1}$ this means that for any $S > 4+\frac{2}{a+1}$ the statement of the Theorem holds. Note that as $a \to \infty$, the speedup sufficient to ensure that the k-th scheduling opportunity occurs in the interval $[\frac{k-1-(a+1)}{r_f}, \frac{k+a}{r_f})$ asymptotically approaches 4. ■

Note finally that for $a = 0$ we obtain the result of the previous Section, i.e. that $S = 6$ suffices to ensure that the k-th scheduling opportunity of any flow f occurs in the interval $[\frac{k-1}{r}, \frac{k}{r})$.

5.1.4 Reducing the Complexity of the Arbiter.

Moving virtual output queues to the Arbiter.

The major drawback of the algorithm described in the previous Section is that the arbiter needs to maintain state for, and perform the arbitration among, all flows traversing the entire switch, which is by far too impractical. We now show how to reduce this complexity by distributing the load of per-flow scheduling among input ports.

The basic idea here is to group flows by the virtual output queues as earlier in Chapter 3, but move the virtual output queue level rate-controllers $\mathbb{S}_q(i)$ to the arbiter. More specifically, the arbiter maintains a logical entry for each input/output pair. These entries contain conceptually the same state information as the rate-controllers $\mathbb{S}_q(i)$ needed in the case of timestamp-based arbitration. In particular, considering FRECF described in the previous Section, the state information for each of the nxm entries at the arbiter will be the rates R_{ij} (which are the same as the rates assigned to the virtual output queues at each input for the timestamp-based arbitration, i.e. R_{ij} is the sum of rates of all flows destined from input i to output j), and the eligibility times s_{ij}. Unlike the previous Section, where FRECF was run by the arbiter *at the flow level* (and consequently the arbiter needed to maintain the rates and the eligibility times for *all* flows), the arbiter considered in this Section maintains and schedules only the nxm logical entries, each of them corresponding to an input/output pair. For convenience, these entries will still be referred to as "virtual output queues" (the quotes indicating that these are the logical queues rather than the actual queues). Just as in the previous Section, during each matching phase the arbiter iteratively computes a maximal matching as follows. It picks the "virtual output queue" corresponding to the input/output pair (i, j) with the fastest rate R_{ij} of all those for which the current time is no less than the current eligibility time s_{ij}, adds it to the matching and removes all "virtual output queues" corresponding to the chosen input and output. It reiterates this process until no more "virtual output queues" can be added to the matching. At the end of the matching phase the arbiter tells the input channels in the chosen matching the output they need to transmit a cell to. If the pair (i, j) is in the current matching, then, once the input i is informed by the arbiter that it can send a cell to output j, the input i invokes the appropriate flow-level scheduler $\mathbb{S}_f(i, j)$, which in turn chooses the cell among all flow queues at this input destined to the output j. Note that just as in the case of the 2-level hierarchical scheduler in the timestamp-based architecture, the schedulers $\mathbb{S}_f(i, j)$ do not operate in real time - the *now* variable of each of these schedulers advances by $\frac{1}{R_{ij}}$ when the scheduler is invoked, while all the state variables remain unchanged when the scheduler is not invoked.

This architecture is shown in Fig. 5-1.

Delay Guarantees.

Just as in the case of the timestamp-based architecture, the delay guarantees that can be ensured in this architecture depend on the properties of input rate-controllers $\mathbb{S}_f(i, j)$ and on the choice of a rate-controller employed at the arbiter. This subsection gives the bound on the switching delay in the case when the flow schedulers $\mathbb{S}_f(i, j)$ at the input channel are RC-WF^{2}Q, while the arbiter employs FRECF. A more general case is considered at the end of this Chapter.

Theorem 27 *If the rate-controllers $\mathbb{S}_f(i, j)$ at the inputs are RC-WF^{2}Q and the arbiter computes the maximal matching using FRECF applied to the nxm virtual output queues then any cell of a flow constrained at the input to the switch by a leaky-bucket (r_f, b) is delivered to the output channel no later than time $\frac{b+4}{r_f} + \frac{1}{S}$ after its arrival to the input channel as long as the speedup $S \geq 6$.*

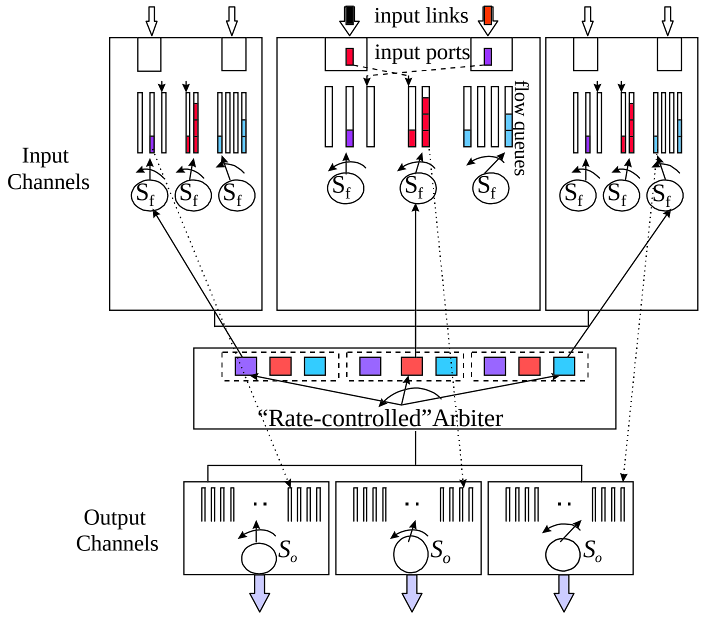

Figure 5-1: Switch architecture with a rate-controlled arbiter. The squares in the arbiter denote the logical entries, one per input/output pair. They are referred to as ”virtual queues”, since they logically correspond to the virtual queues of the timestamp-based architecture.

The proof of this Theorem is based on several lemmas given below. With the exception of several details the proofs of these Lemmas are very similar to the proofs of the corresponding results in the previous Section.

Lemma 28 *In the context of Theorem 27 the k-th arbitration epoch of flow f with assigned rate r_f occurs in the interval $[\frac{k-1}{r_f}, \frac{k+1}{r_f}]$.*

Proof of Lemma 28.

The proof of this Lemma is very similar to that of Theorem 15. By operation of the algorithm, the state variables s_f and f_f for any flow in any of the RC-WF^{2}Q-based schedulers $\mathbb{S}_f(i,j)$ do not depend on the time they are updated, but rather on the sequence number of the update. Therefore, the *sequence* of the values of the state variables in the these schedulers when invoked by the arbiter is indistinguishable from the sequence of an isolated one-level RC-WF^{2}Q scheduler operating in isolation on flows with the same rate assignment on a link of capacity R_{ij}. Consider the *real time* t_k of the k-th scheduling opportunity of some flow f . This time can occur only at the time when the corresponding "virtual output queue" corresponding to input/output pair (i,j) is chosen by the arbiter. Let this be the m-th arbitration epoch of the "virtual output queue" (i,j). By Theorem 20, the m-th scheduling opportunity of any "virtual output queue" occurs in the interval $[\frac{m-1}{R_{ij}}, \frac{m}{R_{ij}}]$. Therefore

$$\frac{m-1}{R_{ij}} \le t_k \le \frac{m}{R_{ij}} \tag{5.3}$$

By operation of the algorithm, the variable τ of $\mathbb{S}_f(i,j)$ at time t_k (at the time the scheduling decision is made, before the update of τ) which we denote as $\tau(t_k)$ is given by

$$\tau(t_k) = \frac{m-1}{R_{ij}} \tag{5.4}$$

Applying Theorem 15 to the $\mathbb{S}_f(i,j)$ operating in isolation on a link of capacity R_{ij} we get

$$\frac{k-1}{r_f} \le \tau(t_k) \le \frac{k}{r_f} \tag{5.5}$$

Further, (5.4) and (5.5) imply

$$\frac{k-1}{r_f}R_{ij} + 1 \le m \le \frac{k}{r_f}R_{ij} + 1$$

which, together with (5.3) yields

$$\frac{k-1}{r_f} \le \frac{m-1}{R_{ij}} \le t_k \le \frac{m}{R_{ij}} \le \frac{k}{r_f} + \frac{1}{R_{ij}} \le \frac{k+1}{r_f}$$

where the last inequality follows from the fact that $R_{ij} \ge r_f$, since R_{ij} is the sum of rates of all flows destined from i to j. Hence, the k-th arbitration epoch of flow f occurs in the interval $[\frac{k-1}{r_f}, \frac{k+1}{r_f}]$. This completes the proof of the Lemma. ■

Lemma 29 *In the context of Theorem 27 a cell arriving at time t for a given flow f with a queue of length Q will be delivered to its output no later than at time $t + \frac{Q+3}{r_f} + \frac{1}{S}$.*

Proof of Lemma 29.

Consider a cell c arriving to the switch at some time t. Let $c_1, c_2 ... c_Q$ be the cells in the queue before c (if c starts a busy period, then $Q = 0$ and there are no cells ahead of it in the queue). Choose k to satisfy $\frac{k-1}{r_f} \le t < \frac{k}{r_f}$. By Lemma 28 it must be that the k-th arbitration epoch of the flow occurs in the interval $[\frac{k-1}{r_f}, \frac{k+1}{r_f}]$, while the $k+1$-st arbitration epoch of this flow occurs in the interval $[\frac{k}{r_f}, \frac{k+2}{r_f}]$. While the first cell in the queue may have missed the k-th arbitration epoch (if the latter occurred in the interval $[\frac{k-1}{r_f}, t)$), it will be scheduled no later than at the $k+1$-st arbitration epoch. Hence, the first cell in the queue at time t must be scheduled to be transmitted to its output no later than by time $\frac{k+2}{r_f}$, which implies that cell c_2 will be scheduled by time $\frac{k+3}{r_f}$, and, inductively, cell c_Q will be scheduled by time $\frac{k+Q+2}{r_f}$, and finally the cell c itself will be scheduled no later than by time $t_{Q+1} = \frac{k+Q+3}{r_f}$. If c started a busy period, then c itself will be scheduled no later than at time t_1, and so will be delivered to its output no later than at time $t_1 + \frac{1}{S} = \frac{k+2}{r_f} + \frac{1}{S} \leq t + \frac{3}{r_f} + \frac{1}{S}$, which proves the statement of the Lemma in this case. Otherwise, the cell c itself will be scheduled by time $t_{Q+1} = \frac{k+Q+2}{r_f} \leq t + \frac{Q+3}{r_f}$, and is guaranteed to arrive to its output by time $t + \frac{Q+3}{r_f} + \frac{1}{S}$. ■

Lemma 30 *If, in the context of Theorem 27, the input traffic of flow f with assigned rate r_f conforms to a leaky bucket (r_f, b) then the queue of this flow is bounded by $b+2$.*

Proof.

The proof is similar to the proof of Corollary 22. Suppose this is not the case. Then there must exist some time t at which there are at least $b+3$ cells of the considered flow in the queue. Let t_0 denote the beginning of the flow's busy period containing t. Since the flow is constrained by a leaky bucket (r_f, b), at most $(t-t_0)r_f + b + \varepsilon r_f$ cells could have arrived to the queue in the interval $[t_0, t+\varepsilon)$ for any $\varepsilon > 0$. On the other hand, by Lemma 28, at least $(t-t_0)r_f - 2$ scheduling opportunities of this flow must have occurred in the interval $[t_0, t] \subset [t_0, t+\varepsilon)$. Furthermore, since the queue has been continuously backlogged in the interval $[t_0, t]$, an actual cell of this flow was scheduled at each of these scheduling opportunities. Therefore, for any $\varepsilon > 0$ the queue is bounded from above $b + 2 + \varepsilon r_f$. Since ε can be chosen arbitrarily small, the statement of the Lemma follows. ■

The proof of Theorem 27 now follows immediately from Lemmas 28 and 30.

Corollary 31 *In the context of Theorem 27 a flow with assigned rate r_f conforms to a leaky bucket $(r_f, 3)$ at the entry to its output channel.*

Proof of Corollary 31.

Consider any interval $[t_1, t_2)$ and let

$$\frac{k-1}{r_f} \leq t_1 < \frac{k}{r_f} \tag{5.6}$$

$$\frac{k+m-1}{r_f} \leq t_2 < \frac{k+m}{r_f} \tag{5.7}$$

for some integer $k \geq 1, m \geq 0$. By Lemma 28 at most $m+2$ cells can be scheduled in $[t_1, t_2)$. From (5.6) $k > t_1 r_f$. From (5.7) $m \leq t_2 r_f - k + 1$. Therefore $m+2 \leq t_2 r_f - k + 3 \leq (t_2 - t1) r_f + 3$. By definition this implies that the flow conforms to a leaky bucket $(r_f, 3)$.■

Corollary 31 in conjunction with Theorem 7 immediately yields

Corollary 32 *In the context of Theorem 27 the output delay of any flow is bounded by* $\frac{\underline{E}^{Out}+3}{r_f}$ *where* $\underline{E}^{Out}$ *is the lower bound on the work discrepancy of the output scheduler.*

Finally, the following Theorem gives an upper bound on the total switching delay:

Theorem 33 *The total switching delay of any cell of a flow constrained by a leaky-bucket* (r_f, b) *at the entry to the switch with speedup* $S \geq 6$, *with RC-WF*^{2}*Q at the input channels and a FRECF-based arbiter is bounded by* $\frac{\underline{E}^{Out}+b+8}{r_f} + \frac{1}{S} + \frac{1}{C_{out}}$ *where* C_{out} *is the speed of the output link.*

This Theorem is proved simply by adding the bounds given by Lemma 28 and Corollary 32 and adding an additional $\frac{1}{C_{out}}$ to account for the time required to transmit one cell at the speed of the outgoing link.

Speedup Required for Deterministic Delay Guarantees for Fractional Link Utilization.

The results of the previous Section hold for up to 100% booking of channel bandwidth by guaranteed flows. It will now be shown that if the total bandwidth allocated to guaranteed flows does not exceed a fraction $0 < \alpha \leq 1$ of the capacity of any channel, then the speedup $S \geq 6\alpha$ suffices to ensure deterministic bandwidth and delay guarantees. For example, as long as at most half the bandwidth of any channel is allocated to guaranteed traffic, speedup $S \geq 3$ suffices to ensure delay guarantees of the previous Section.

This result is based on a straightforward generalization of Theorem 20:

Theorem 34 *If the total bandwidth allocated to guaranteed flows sharing any channel does not exceed a fraction* $0 < \alpha \leq 1$, *then with speedup* $S \geq 6\alpha$ *the* k*-th scheduling opportunity of flow* f *with rate* r_f *under FRECF occurs in the interval* $[\frac{k-1}{r_f}, \frac{k}{r_f}]$.

Proof. The proof is almost identical to that of Theorem 20. Just as in the case of RC-WF^{2}Q, it is convenient to imagine that the scheduler is backlogged with "dummy" cells which are replaced by real cells upon their arrival. We first show that the first scheduling opportunity of any flow f occurs in the interval $[0, \frac{1}{r_f}]$ for any $S \geq 4$. Note that the first cell of any flow becomes eligible at time 0. Consider any flow f destined from input i to output j and suppose that its first cell c is not scheduled until after time $\frac{1}{r_f}$. By the operation of the algorithm this means that at each matching phase an eligible cell either from input i or destined to output j with $r_{f'} \geq r_f$ was scheduled by the arbiter. Note now that for any such flow $f^{'}$, only cells with eligibility times in the interval $[0, \frac{1}{r_f}]$ can prevent c from being scheduled in the interval $[0, \frac{1}{r_f}]$. It is easy to see that for any $r_{f'} \geq r_f$ there could be at most $\frac{r_{f'}}{r_f} + 1 \leq 2\frac{r_{f'}}{r_f}$ of such cells. Recalling that the sum of rates of all flows at any input (or destined to any output) does not exceed α, it follows that there are at most $\sum_{f'} 2\frac{r_{f'}}{r_f} \leq \frac{2\alpha}{r_f}$ cells with eligibility time not exceeding $\frac{1}{r_f}$ at input i, which includes c itself. Therefore, the maximum amount of c's competition at the input is at most $\frac{2\alpha}{r_f} - 1$, and similarly, there are at most $\frac{2\alpha}{r_f} - 1$ competing cells destined to output j. Therefore there are at most $\frac{4\alpha}{r_f} - 2$ cells that can prevent c from being scheduled in the interval $[0, \frac{1}{r_f}]$. On the other hand, by Lemma 4 there are at least $\frac{S}{r_f} - 1$ matching phase boundaries in the interval $[0, \frac{1}{r_f}]$, and hence there will not be enough competing cells for any $S \geq 4\alpha$.

We now show that for $S \geq 6\alpha$ the statement of the Theorem holds for any $k > 1$ as well. Note that by operation of the algorithm a cell cannot be scheduled before its eligibility time. Therefore, we only need to prove that the k-th cell of each flow is scheduled before $\frac{k}{r_f}$, which is its ideal finishing time in fluid. Suppose that the statement of the Theorem is false. This means that there exists some flow f and some $k \geq 2$ such that the k-th cell of f was not scheduled by $\frac{k}{r_f}$. We call such cell a violating cell. Consider the violating cell c with the smallest eligibility time of all violating cells, and let f be the flow it belongs to. Assume that f is destined from an input i to an output j. Let k be c's index, so that its eligibility time is $\frac{k-1}{r_f}$. In order for c to not be transmitted by its deadline $\frac{k}{r_f}$, it must be that at each matching phase boundary in the interval $[\frac{k-1}{r_f}, \frac{k}{r_f}]$ an eligible cell with faster rate was scheduled either from the input i, and/or to the output j. Note now that by the choice of c, it must be that all cells with ideal finish times less than $\frac{k-1}{r_f}$ must have been transmitted by $\frac{k-1}{r_f}$. Therefore, the only "competition" to c that can prevent it from being scheduled by its deadline $\frac{k}{r_f}$ are those cells whose rates are larger, whose eligibility time does not exceed $\frac{k}{r_f}$, and whose finish times are at least $\frac{k-1}{r_f}$. It is easy to see that for any $r_{f'} \geq r_f$ there could be at most $\frac{r_{f'}}{r_f} + 2 \leq 3\frac{r_{f'}}{r_f}$ of such cells. Summing these cells as before over all flows with rates $r_{f'} \geq r_f$ at c's input and output, and recalling that the sum of rates of all flows sharing a particular channel does not exceed α, we see that the maximum amount of competition that can prevent c from being scheduled by $\frac{k-1}{r_f}$ is at most $\frac{6\alpha}{r_f} - 2$, which falls short of the number of matching phase boundaries in the interval $[\frac{k-1}{r_f}, \frac{k}{r_f}]$, which is at least $\frac{6\alpha}{r_f} - 1$. The obtained contradiction completes the proof of the Theorem. ■

The main result of this Section is now given by the following Theorem, the proof of which is identical to the proof of the corresponding result of the switching delay in the previous Section given by Theorem 34, and is therefore omitted here.

Theorem 35 *If the total bandwidth allocated to guaranteed flows sharing any channel does not exceed a fraction* $0 < \alpha \leq 1$,*the total switching delay of any cell of a flow constrained by a leaky-bucket* (r_f, b) *at the entry to the switch with speedup* $S \geq 6\alpha$, *with RC-WF*^{2}*Q at the input channels and a FRECF-based arbiter is bounded by* $\frac{E^{Out}+b+8}{r_f} + \frac{1}{S} + \frac{1}{C_{out}}$ *where* C_{out} *is the speed of the output link.*

Some Implementation-Related Issues.

While the details of implementation are beyond the scope of this dissertation, this Section contains a brief discussion of the key issues related to implementation.

It is important to note that in the framework discussed in the previous two subsections the amount of control communication between the input/output channels and the arbiter is relatively small. When a new flow enters or leaves the switch, the arbiter needs to be notified about the change in the total rate assigned to the appropriate input/output pair. Hence, the communication between the arbiter and the input channels is still required at the time of connection setup. It should be noted that for connection-oriented guaranteed flows it is expected that the duration of a connection is much larger than the set-up/tear-down time, and so the efficient operation during normal operation is highly desirable. For the period of time when the number and the rates of all flows remains stable, the arbiter described in this Chapter does not need any information from the inputs to make its decisions since it maintains all the scheduling variables locally.

Note that since the arbiter is rate-controlled, it does not need to know whether a particular input/output pair actually has a cell to transmit. If it chooses an input/output pair which currently has no cells to send, then the cell is simply not sent[2]. This is precisely what allows the reduction of the communication overhead, which becomes a substantial bottleneck in high speed switches. For example, the communication overhead is a major obstacle for the algorithm described in [7].

Essentially, the only limitation of the approach described in this Chapter is the speed of the arbiter. In the straightforward sequential implementation the complexity per matching phase of the arbiter is $O(n\text{x}m)$, where n and m are the number of input and output channels in the switch. Note that all known maximal matching algorithms have the same order of worst case complexity.

As discussed in the Introduction, the rate of increase in processing speeds appears to be far ahead of the rate of the increase in memory speeds. Therefore, shifting the memory speed bottleneck to processing speed appears to be a substantial advancement.

[2]Note that such "missed" opportunities have no effect on delay and bandwidth guarantees of any other flows. Nevertheless, at first glance it may seem that this may cause unnecessary loss of bandwidth. However, this becomes less of a concern if the rate-controlled arbitration is used to schedule a subset of guaranteed flows, while best-effort traffic is scheduled at lower priority using some simpler mechanism. The interaction of guaranteed and best-effort traffic will be discussed in more detail in the next chapter.

5.1.5 Using Other Rate-Controllers at the Arbiter.

So far the rate-controller employed at the arbiter was assumed to be FRECF. For this case it could be shown that, with the appropriate value of speedup, it is guaranteed that each input/output pair receives its arbitration opportunities with the appropriate frequency with a very small discrepancy from the ideal rate corresponding to this input/output pair. In the context of this Chapter the accuracy of the arbiter with respect to each input/output pair has been expressed in terms of the discrepancy between the time of the actual arbitration opportunities and the ideal transmission time of the aggregate flow between any given input/output pair. More specifically, it was shown that if the total rate of flows sharing an input/output pair (i, j) is R_{ij}, then the k-th arbitration opportunity of this input/output pair occurs in the interval $[\frac{k-1}{R_{ij}}, \frac{k}{R_{ij}}]$, while the ideal time of the k-th arbitration opportunity is $\frac{k-1}{R_{ij}}$.

In principle, any other rate-controller can be incorporated into the arbiter. The accuracy of this rate-controller will affect the switching delays. In fact, using an argument almost identical to the proof of switching delay for FRECF in Section 5.1.4, it will now be shown that

Theorem 36 *If the arbiter guarantees that the k-th arbitration opportunity of input/output pair occurs in the interval $[\frac{k-A-1}{R_{ij}}, \frac{k+B}{R_{ij}}]$, (where $A \geq 0$, $B \geq 0$, are constants), and if the inputs employ RC-WF^{2}Q-based flow schedulers $\mathbb{S}_f$, then the total switching delay of a flow constrained by a leaky bucket (r_f, b) is upper bounded by $\frac{\underline{E}^{Out}+b+3(A+B)+7}{r_f} + \frac{1}{S} + \frac{1}{C_{out}}$*
.

Here the constants A and B characterize the accuracy of the arbiter in terms of the discrepancy between the times of the ideal and the actual arbitration opportunities of any given input/output pair. The proof of the Theorem is based on several lemmas.

Lemma 37 *In the context of Theorem 36, the k-th arbitration opportunity of flow f with assigned rate r_f occurs in the interval $[\frac{k-A-1}{R_{ij}}, \frac{k+B+1}{R_{ij}}]$*

Proof of Lemma 37.

By the operation of the algorithm the state variables s_f and f_f for any flow in any of the RC-WF^{2}Q-based schedulers $\mathbb{S}_f(i,j)$ do not depend on the time they are updated, but rather on the sequence number of the update. Therefore, the *sequence* of values of the state variables in these schedulers when invoked by the arbiter is indistinguishable from the sequence of an isolated one-level RC-WF^{2}Q scheduler operating in isolation on flows with the same rate assignment on a link of capacity R_{ij}. Consider the *real time* t_k of the k-th scheduling opportunity of some flow f . This time can occur only at the time when the corresponding "virtual output queue" corresponding to input/output pair (i,j) is chosen by the arbiter. Let this be the m-th arbitration opportunity of the "virtual output queue" (i,j). By the statement of the Theorem, the m-th scheduling opportunity of any "virtual output queue" occurs in the interval $[\frac{m-A-1}{R_{ij}}, \frac{m+B}{R_{ij}}]$. Therefore

$$\frac{m-A-1}{R_{ij}} \leq t_k \leq \frac{m+B}{R_{ij}} \tag{5.8}$$

By operation of the algorithm, the variable τ of $\mathbb{S}_f(i,j)$ at time t_k (at the time the scheduling decision is made, before the update of τ) which we denote as $\tau(t_k)$ is given by

$$\tau(t_k) = \frac{m-1}{R_{ij}} \tag{5.9}$$

Applying Theorem 15 to the $\mathbb{S}_f(i,j)$ operating in isolation on a link of capacity R_{ij} we get

$$\frac{k-1}{r_f} \leq \tau(t_k) \leq \frac{k}{r_f} \tag{5.10}$$

(5.9) and (5.10) imply

$$\frac{k-1}{r_f}R_{ij} + 1 \leq m \leq \frac{k}{r_f}R_{ij} + 1$$

which, together with (5.8) yields

$$\frac{k-1-A}{r_f} \leq \frac{k-1}{r_f} - \frac{A}{R_{ij}} \leq \frac{m-A-1}{R_{ij}} \leq t_k \leq \frac{m+B}{R_{ij}} \leq \frac{k}{r_f} + \frac{1}{R_{ij}} + \frac{B}{R_{ij}} \leq \frac{k+1+B}{r_f}$$

where the first and the last inequalities follow from the fact that $R_{ij} \geq r_f$, since R_{ij} is the sum of rates of all flows destined from i to j. Hence, the k-th arbitration opportunity of flow f occurs in the interval $[\frac{k-1-A}{r_f}, \frac{k+1+B}{r_f}]$. This completes the proof of the Lemma.■

Lemma 38 *In the context of Theorem 36, a cell arriving at time t and finding a queue of length Q will be delivered to its output no later than by time $t + \frac{Q+A+B+3}{r_f} + \frac{1}{S}$.*

Proof of Lemma 38.

Consider a cell c arriving to the switch at some time t. Let $c_1, c_2 ... c_Q$ be the cells in the queue before c (if c starts a busy period, then $Q = 0$ and there are no cells ahead of it in the queue). Let the arrival time t of the cell c satisfy $\frac{k-1}{r_f} \leq t < \frac{k}{r_f}$ for some $k \geq 1$. By the previous Lemma it must be that the next arbitration opportunity of the flow occurs no later than in the interval $[\frac{k}{r_f}, \frac{k+A+B+2}{r_f}]$. Hence, the first cell in the queue at time t must be scheduled to be transmitted to its output no later than by $t_1 = \frac{k+A+B+2}{r_f}$, which implies that cell c_2 will be scheduled by time $t_2 = \frac{k+A+B+2}{r_f}$, and, inductively, cell c_Q will be scheduled by time $t_Q = \frac{k+A+B+Q+1}{r_f}$, and finally the cell c itself will be scheduled no later than by time $t_{Q+1} = \frac{k+A+B+Q+2}{r_f}$. If c started a busy period, then c itself will be scheduled no later than at time t_1, and so will be delivered to its output no later than at time $t_1 + \frac{1}{S} = \frac{k+A+B+2}{r_f} + \frac{1}{S} \leq t + \frac{A+B+3}{r_f} + \frac{1}{S}$, which proves the statement of the Lemma in this case. Otherwise, as discussed above, the cell c will be scheduled by time $t_{Q+1} = \frac{k+Q+A+B+2}{r_f} \leq t + \frac{Q+A+B+3}{r_f}$, and is guaranteed to arrive to its output by time $t + \frac{Q+A+B+3}{r_f} + \frac{1}{S}$. ■

Lemma 39 *If, in the context of Theorem 36, the input traffic of flow f with assigned rate r_f conforms to a leaky bucket (r_f, b) then the queue of this flow is bounded by $b+A+B+2$.*

Proof of Lemma 39.

Suppose this is not the case. Then there must exist some time t at which there are at least $b + A + B + 1$ cells of the considered flow in the queue. Let t_0 denote the beginning of the flow's busy period containing t.

Since the flow is constrained by a leaky bucket (r_f, b), at most $(t-t_0)r_f+b+\varepsilon r_f$ cells could have arrived to the queue in the interval $[t_0, t+\varepsilon)$ for any $\varepsilon > 0$. On the other hand, by Lemma 37, at least $(t-t_0)r_f - A - B - 2$ scheduling opportunities of this flow must have occurred in the interval $[t_0, t] \subset [t_0, t+\varepsilon)$. Furthermore, since the queue has been continuously backlogged in the interval $[t_0, t]$, a cell of this flow was actually transmitted at each of these scheduling opportunities. Therefore, for any $\varepsilon > 0$ the queue is bounded from above by $(t-t_0)r_f + b + \varepsilon r_f - (t-t_0)r_f + A + B + 2 = b + A + B + 2$. Since ε can be chosen arbitrarily small, the statement of the Lemma follows. ■

Lemma 40 *In the context of Theorem 36 a flow with assigned rate r_f conforms to a leaky bucket $(r_f, A+B+3)$ at the entry to its output channel.*

Proof of Lemma 40.

Consider any interval $[t_1, t_2)$ and let

$$\frac{k-1}{r_f} \leq t_1 < \frac{k}{r_f} \tag{5.11}$$

$$\frac{k+m-1}{r_f} \leq t_2 < \frac{k+m}{r_f} \tag{5.12}$$

for some integer $k \geq 1, m \geq 0$. By Lemma 37 at most $m + A + B + 2$ cells can be scheduled in $[t_1, t_2)$. From (5.11) $k > t_1 r_f$. From (5.12) $m \leq t_2 r_f - k + 1$. Therefore $m + A + B + 2 \leq t_2 r_f - k + A + B + 3 \leq (t_2 - t1) r_f + A + B + 3$. By definition this implies that the flow conforms to a leaky bucket $(r_f, A+B+3)$. ■

Proof of Theorem 36.

First note that Lemmas 37 and 39 it immediately follows that any cell of a flow constrained by a leaky bucket (r_f, b) at the entry to the switch is delivered to the output channel no later than time $\frac{b+2(A+B)+5}{r_f} + \frac{1}{S}$ after its arrival. Using Lemma 40 in conjunction with Theorem 7 immediately implies that the output delay of any flow is bounded by $\frac{\underline{E}^{Out}+A+B+3}{r_f}$ where $\underline{E}^{Out}$ is the lower bound on the work discrepancy of the output scheduler. Adding these bounds and adding further an additional $\frac{1}{C_{out}}$ to account for the time required to transmit one cell at the speed of the outgoing link we obtain that the total switching delay is bounded by $\frac{\underline{E}^{Out}+b+3(A+B)+8}{r_f} + \frac{1}{S} + \frac{1}{C_{out}}$. ■

Chapter 6

Providing For Different Classes of Service

6.1 Coexistence of Guaranteed Flows with Other Classes of Service

So far the main emphasis of this thesis has been on providing bandwidth and delay guarantees to the so-called guaranteed flows. However, the results of the previous Chapters can also be applied to allow support for flows with less stringent QoS requirements along with guaranteed flows. In particular, it is possible to ensure that while the guaranteed flows receive the guarantees promised at connection setup, the "lower grade service" traffic fills up the remaining bandwidth, ensuring high link utilization. This Section discusses this issue in more detail.

Consider the switch architecture of Chapter 5 (Figure 5-1), where the arbiter maintains a logical entry per virtual output queue and uses some rate-controlled arbitration mechanism such as FRECF to compute a maximal matching. Recall that FRECF computes a maximal matching only *among eligible logical virtual output queues.* Therefore, although no more eligible queues can be added to the matching, there may still be some unmatched inputs and outputs for which no connections have been made. If there is any "lower grade" traffic, it may be possible to transmit some number of the "lower grade" cells between these unmatched inputs and outputs. This observation motivates the following approach. Suppose that the switch runs two different algorithms $\mathcal{A}^G$ and $\mathcal{A}^{LG}$ for guaranteed and "lower grade" traffic respectively. Assume that $\mathcal{A}^G$ is run at strictly higher priority than $\mathcal{A}^{LG}$, and suppose for example that $\mathcal{A}^G$ is FRECF, while $\mathcal{A}^{LG}$ is *any* arbitration algorithm which computes a maximal matching on a bi-partite graph given a set of requests from a subset of inputs to a subset of outputs. Examples of such algorithms can be SLIP [17], PIM [1], WPIM [24], LOOFA[19], etc. The arbitration proceeds as follows.

1. At the beginning of a matching phase, the arbiter runs $\mathcal{A}^G$ and computes a maximal matching among all eligible logical virtual output queues. At the end of this computation a subset of inputs and outputs are matched. Let $\mathcal{S}_I^G$ and $\mathcal{S}_O^G$ be the subsets of all inputs and outputs matched by $\mathcal{A}^G$, and let $\mathcal{S}_I$ and $\mathcal{S}_O$ be the sets of remaining inputs and outputs.

2. For the chosen input/output pairs in sets $\mathcal{S}_I^G$ and $\mathcal{S}_O^G$ the flow-level input schedulers (e.g. WF^{2}Q as described in the previous Chapter) are used to choose guaranteed flows corresponding to the chosen "guaranteed matching". If the chosen guaranteed flow's queue is non-empty, its HOL cell is transmitted. If for any input/output pair (i, j) the chosen guaranteed flow queue is empty, the input i and the output j are added to the so far unmatched sets $\mathcal{S}_I$ and $\mathcal{S}_O$.

3. Once all guaranteed cells chosen in the previous step are transmitted, $\mathcal{A}^{LG}$ is invoked to compute a maximal matching among the remaining unmatched sets of inputs and outputs $\mathcal{S}_I$ and $\mathcal{S}_O$, choosing some subsets $\mathcal{S}_I^{LG} \subseteq \mathcal{S}_I$ and $\mathcal{S}_O^{LG} \subseteq \mathcal{S}_O$ for the "lower-grade matching".

Note here that unlike $\mathcal{A}^G$ which in the described context operates independently of the input schedulers, $\mathcal{A}^{LG}$ in principle may involve some additional communication between inputs, outputs and the arbiter, as well as some additional scheduling of "lower grade" traffic at the input channels. For example, the round-robin request/grant protocol described in the example at the beginning of Chapter 2 involves exchanging request and grants, as well as round-robin scheduling at the inputs. Similar control communication and scheduling is used in algorithms such as SLIP, PIM, etc. that could be chosen for this purpose.

It is easy to see that regardless of the choice of $\mathcal{A}^{LG}$, the arbitration of guaranteed flows in this framework *in the presence of "lower grade" traffic* is indistinguishable from the case considered in Chapter 5 i*n the absence of "lower grade" traffic.* This is a simple consequence of the fact that guaranteed flows are treated at a strictly higher priority and therefore are completely unaffected by the presence of the "lower grade" traffic. Hence the framework described here preserves all the guarantees shown in Chapter 5 for the guaranteed flows.

Recall that the results of Chapter 2 imply that as long as traffic is restricted by some traffic management algorithm so that the total bandwidth consumed by it does not exceed the capacity available to it, (in the sense that in any interval of time of length t the total amount of input traffic sharing a channel of capacity C does not exceed $Ct + B$ for some constant B), then any maximal matching algorithm can ensure a 100% bandwidth guarantee with the appropriate speedup. Note now that the two-level arbitration algorithm described in this Section (e.g. apply $\mathcal{A}^G$ first, then apply $\mathcal{A}^{LG}$ on remaining inputs and outputs) yields a maximal matching. Assuming the existence of some traffic management algorithm which ensures that there exists some constant[1] B^{LG} such that the amount of input "lower grade" traffic $A(t_1, t_2)$ arriving in any interval (t_1, t_2) never exceeds the limit

$$A(t_1, t_2) \leq C^{LG}(t_2 - t_1) + B^{LG} \tag{6.1}$$

[1]Recall from the discussion in chapter 2 that this constant essentially determines the buffer requirements needed to ensure that lower traffic suffers no loss. An example of a service class for which this assumption holds is Available Bit Rate (ABR) service in ATM.

(where $C^{LG} = C - C^G$ is the bandwidth available to best effort flows), and assuming that the guaranteed flows are leaky-bucket constrained at the entry to the switch, the results of Chapter 2 then immediately imply that the "lower grade" traffic can be provided a 100% bandwidth guarantee as long as the speedup satisfies $S \geq 4$. In conjunction with the results of Chapter 5 this means that $S \geq 6$ suffices to ensure that, with FRECF used for $\mathcal{A}^G$, *any* maximal matching arbitration algorithm guarantees that "lower grade" traffic can achieve 100% of the "leftover" bandwidth C^{LG}. This assumes the existence of a traffic management algorithm controlling the input rates of the "lower grade" traffic as discussed above.

In fact, using the results of Chapter 5 related to the speedup needed to ensure delay guarantees in a fractionally booked system, a stronger statement can be made in this context if the admission control policy for guaranteed flows ensures that $C^G \leq \frac{2}{3}C$, i.e. the guaranteed flows are not allowed to book more than two thirds of any channel capacity. In this case using FRECF-based $\mathcal{A}^G$ with *any* maximal matching algorithm $\mathcal{A}^{LG}$ and the speedup $S = 4$ suffices to ensure both deterministic bandwidth and delay guarantees for guaranteed traffic as well as bandwidth guarantees for "lower grade" traffic, provided the latter is controlled by a traffic management algorithm to satisfy (6.1). Furthermore, if the OCF-based algorithm discussed in Chapter 2 is used for "lower grade" traffic scheduler $\mathcal{A}^{LG}$, then the results of Sections 2.3 and Chapter 5 imply that the speedup $S = 2$ suffices to provide such guarantees to both guaranteed traffic and "lower grade" traffic as long as the bandwidth allocated for guaranteed flows does not exceed half of any channel bandwidth.

Note finally that a very similar approach can be used if timestamp-based arbitration discussed in Chapters 3-4 is used for $\mathcal{A}^G$. Although the delay bounds obtained for guaranteed flows in this framework will be worse than that for FRECF (since they depend on the size of the switch), the results of Chapters 3 and 4 imply that $S = 4$ suffices in this case to ensure that the "lower grade" traffic can be provided the bandwidth guarantee (as long as it does not exceed the bandwidth unused by the guaranteed flows), while preserving the bandwidth and delay guarantees for the guaranteed traffic as described in Chapters 3-4 even if guaranteed traffic occupies a high percentage of some channel bandwidth.

6.2 Providing Fair Service for "lower grade" Flows

In the previous Section it was assumed that "lower grade" traffic is managed to ensure that the total input rate of "lower grade" traffic does not exceed the available capacity in the sense that given sufficient buffer, no data is lost. This assumption, however, did not account for potential preferential treatment of some "lower grade" flows at the expense of other best effort traffic. It is desirable that given identical demands, flows sharing the same channel/link should be given identical service by the switch.

In practice, the share of service guaranteed to a flow strongly depends on the buffering and scheduling policies employed by the switch. In the context of the crossbar architecture with virtual output queueing at the inputs, providing per-flow fair service turns out to be a significant challenge. Many of the simple arbitration algorithms used in the industry such as SLIP [17], PIM [1] attempt to treat *each virtual output queue* fairly. However, this may result in unfairness towards individual flows. Suppose for example that input 1 has 100 flows destined to output 1, while input 2 has only one flow destined to this output. As a result, the virtual output queue at input 1 destined to output 1 will represent 100 flows, while the corresponding virtual output queue at input 2 will represent a single flow. Clearly, if these two virtual output queues are ensured the same service rate, then each of the flows at input 1 will receive only 0.01 of the service received by the flow at input 2. Note that the flow from input 2 will be given half the capacity of the output channel, while each of the other flows will be given only $\frac{1}{200}$ of this capacity.

One way of correcting this problem is to make the arbitration mechanism aware of the rates at which the "lower grade" virtual output queues must be served. For example, suppose that each flow is assigned a rate of $min(\frac{C_{Inp}}{N_{inp}}, \frac{C_{Out}}{N_{Out}})$, where C_{Inp} and C_{Out} denote the amounts of bandwidth available for "lower grade" traffic at the input and output channels respectively, while N_{inp} and N_{Out} denote the numbers of "lower grade" flows sharing the input and the output. Once this is done, one can assign each "lower grade" virtual output queue at each input the rate which is simply the sum of fair rates of all "lower grade" flows. Now, we can use any of the algorithms described in Chapters 3-5 to arbitrate among the "lower grade" virtual output queues as well. Since "lower grade" traffic does not require strict delay guarantees, one can use a simple round-robin scheduler for flow-level schedulers at the input. For example, in the absence of guaranteed flows one can use the same architecture as shown in Figure 5-1, where the rate-based arbitration described in Chapter 5 is used, where the flow-level schedulers at each input are round-robin schedulers. It is easy to see that the results of Chapter 5 imply that *in the absence of guaranteed traffic* each virtual output queue is ensured the service equal to $R^{LG}_{ij} = \sum_{i=1}^{N_{Inp}} min(\frac{C_{Inp}}{N_{inp}}, \frac{C_{Out}}{N_{Out}})$, and so each virtual output queue will be given enough service to ensure that each flow can receive its fair service. This architecture does not consider the coexistence of "lower grade" traffic with guaranteed flows. It turns out that once the "fair" rates of "lower grade" traffic are estimated, the following simple mechanism can be used. Consider the architecture shown in Figure 6-1, where the "lower grade" and the guaranteed arbitration coexist.

We use the arbitration mechanism of Chapter 5, where the arbiter maintains mxn logical virtual output queue entries, while the inputs run WF^{2}Q flow schedulers for guaranteed traffic. However, the rates assigned to a logical virtual output queue Q_{ij} are now $R_{ij} = R^G_{ij} + R^{LG}_{ij}$. The arbitration between the logical virtual output queues proceeds exactly as described in Chapter 5, e.g. using FRECF. At the input channels the guaranteed flow queues are still grouped by the output as in Chapter 5, except now there is an additional "dummy" queue Q^D_{ij} per each output. The dummy queue Q^D_{ij} is assigned the R^{LG}_{ij}, which is the combined estimated rate of all "lower grade" flows destined from input i to output j. Each Q^D_{ij} is scheduled along with other guaranteed flow queues by the guaranteed scheduler $\mathbb{S}_f(i, j)$ which is invoked any time the arbiter

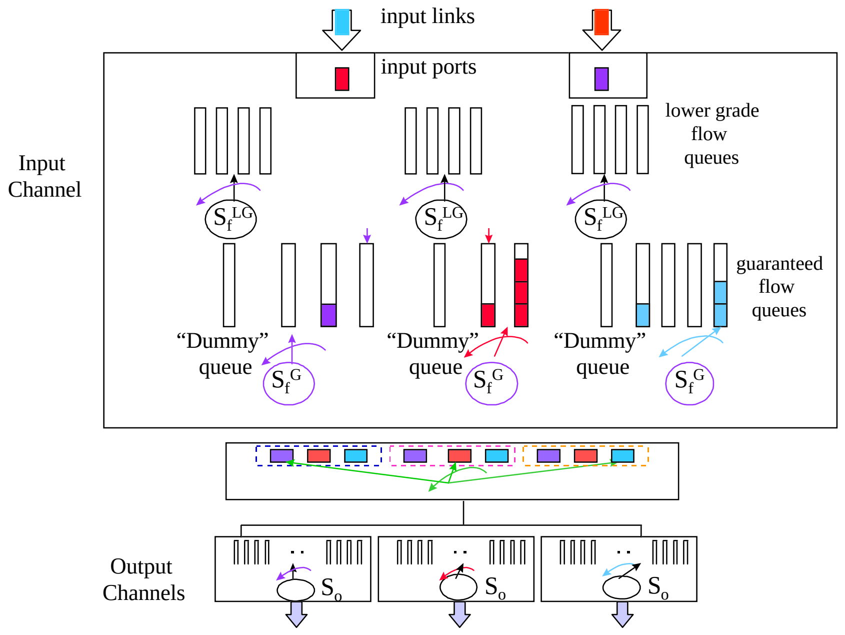

Figure 6-1: Architecture for combined scheduling of "lower grade" and guaranteed flows.

chooses the input/output pair (i, j). Whenever scheduler $\mathbb{S}_f(i, j)$ chooses the "dummy" queue, a round-robin scheduler $\mathbb{S}^{LG}(i, j)$ is invoked to pick the next non-empty queue in its round-robin schedule among the "lower grade" flows at this input destined to output j. Whenever a "regular" guaranteed flow is scheduled and the queue of this flow is non-empty, the HOL cell of that guaranteed flow is transmitted. However, if the chosen guaranteed flow has no cells to send, this scheduling opportunity is "passed" to the round-robin scheduler $\mathbb{S}^{LG}(i, j)$ to choose the next non-empty "lower grade" queue. Note here that when $\mathbb{S}_f(i, j)$ chooses the dummy queue Q^D_{ij}, the state variables of the dummy queue are updated according to the operation of the scheduler. In contrast, when a *real flow* queue is chosen, but has no cells to send, it is the state variables of that real queue which are updated, whereas the state of the dummy queue remains the same.

It is easy to see that the results of Chapter 5 immediately imply that the bandwidth and delay guarantees of all guaranteed flows remain as derived there. Moreover, each "dummy" queue corresponding to the "lower grade" traffic is guaranteed to be scheduled at least at the correct "fair" rate R_{ij}^{LG}. The round-robin "lower grade" scheduler in turn ensures that all busy best-effort queues are given an equal share of the total "lower grade" service rate. Hence, the approach described here eliminates the problem of unfair treatment of some flows at the expense of other flows.

A drawback of the approach presented here for "lower grade" traffic is that it is non-workconserving. Although scheduling opportunities unused by guaranteed flows are passed to "lower grade" flows, it is still possible that when a particular input-output pair is chosen by the matching algorithm, there may be no "lower grade" traffic between this input and output at that time. Of course, this problem can be eliminated by running a second round of "lower grade" arbitration using yet another "lower grade" algorithm and run it at lower priority as in the previous Section to choose a maximal matching among a subset of inputs and outputs for which no cells were transmitted in the first round. However, this clearly adds yet more complexity.

Another drawback in the described approach lies in the estimation of fair share as simply the minimum of an equal share of the input and output bandwidth. This approach does not take into account the fact that some flows may not be able to use this computed fair share, while other flows may benefit by sharing the portion of bandwidth unused by such flows. A definition of fairness that captures this notion is the so-called maxmin fairness (see for example [5]). Let F be the set of all ("lower grade") flows in the switch and let r_f denote the input rate (demand) of flow f. Assuming that all demands r_f are known, the maxmin fair bandwidth allocation in the context of a crossbar switch can be *defined* by the following iterative procedure:

1. For each channel i with $N_i \neq 0$ compute $R_i = \frac{C_i}{N_i}$, where C_i is the available capacity of channel i, and N_i is the number of "lower grade" flows sharing this channel

2. Find $min_{f \in F, i}(r_f, R_i)$.

3. If the minimum in the previous step is r_f and is achieved for some flow(s) f, then do the following *for all such flows*:

   (a) if i, j are the input and output channels of f respectively, set $C_i \leftarrow C_i - r_f$, $N_i \leftarrow N_i - 1$

   (b) assign the flow f its demand r_f and remove f from consideration, i.e. set $F \leftarrow F \backslash \{f\}$

4. If the minimum in step 2 is R_i and achieved for some channel i, then for all flows f sharing this channel do the following:

   (a) if i, j are the input and output channels of f respectively, set $C_i \leftarrow C_i - R_i$, $N_i \leftarrow N_i - 1$

   (b) assign the flow f rate R_f and remove f from consideration, i.e. set $F \leftarrow F \backslash \{f\}$

5. If $F \neq \emptyset$, go to step 1; otherwise stop

The rate allocation obtained by this procedure is called a maxmin fair allocation. While it captures the desire to allocate to each flow no more bandwidth than it can use while preserving the equal sharing of bottleneck capacity, computing this allocation is undoubtedly more complicated than a simple equal share described earlier in this Section. In addition to extra computational complexity, it requires the knowledge of the input rates (demands) of all flows. While in some cases this information may be readily available (for example in the Available Bit Rate (ABR) service in ATM networks the rate information is explicitly written in special resource management (RM) cells), in a typical packet switching network the input rates of flows are not known. To obtain this information, one would need to perform per-flow rate measurement at the input to the switch, which is typically expensive and difficult to perform at high speeds. Hence, there is an obvious trade-off between the complexity of the computation of fair share and the accuracy of this computation.

Chapter 7

Summary and Discussion

7.1 Summary of the Contributions.

This dissertation has investigated a number of issues related to providing various degrees of QoS in a crossbar switch architecture. To the best of my knowledge, the results presented here were the first to demonstrate that it is possible to provide deterministic delay guarantees in crossbars with speedup which is independent of the size of the switch. While recently there have been new results which allow providing even stricter delay guarantees with lower speedup values in crossbars by strictly emulating an output buffered switch, the complexity of implementation of these algorithms appears to be too high for practical high speed implementations (see the next Section for a more detailed discussion of this issue). As a result, it appears that the algorithms for providing delay guarantees described in this dissertation remain the only known algorithms that yield practical high speed implementations while also providing strict delay guarantees.

It has been shown that the architecture which suffices to provide deterministic delay guarantees to leaky-bucket constrained flows can be decomposed into three fairly independent pieces - the input and output channel schedulers and the arbiter. It has been shown that while these pieces can be designed independently of each other, they interact in a predictable way, allowing the computation of delay guarantees resulting from the properties of the individual pieces. Thus, while allowing independent design of the building blocks of the architecture, the results of this thesis also provide means for a quantitative understanding of how implementation trade-offs for any particular building block affect the guarantees that can be provided by the whole architecture.

Another contribution of this dissertation has been in demonstrating that only a limited amount of speedup ($S = 4$) suffices to overcome the bandwidth loss for an arbitrary maximal matching arbitration algorithm, while also demonstrating that there exist maximal matching algorithms which yield the same bandwidth guarantees with lower speedup such as ($S = 2$). In particular, this result implies that the bandwidth loss due to arbitration conflict in an *arbitrary maximal matching arbitration* algorithm does not depend on the size of the switch, and can be overcome by increasing the speedup of the switch fabric by a small constant factor.

In general, the results of this dissertation suggest that, contrary to the widely accepted view that crossbars are unsuitable for providing QoS guarantees in Integrated Services Networks, this architecture is capable of providing strict bandwidth and delay guarantees as long as the switch fabric has a limited speedup, and an appropriate scheduling architecture is used. Since the crossbar architecture is widely considered the most scalable in required memory speed, the existence of implementable QoS-capable algorithms for this architecture open the way for practical high-speed crossbar implementations for modern high-speed Integrated Services Networks.

7.2 Relationship To the Latest Results in Providing Delay Guarantees In Crossbar Architectures.

As mentioned earlier in this dissertation, it has been recently reported that it is possible to strictly emulate an output buffered switch with a WFQ-like scheduler at the outputs in a crossbar with speedup $S = 2$. The results of this work imply that for flows which are leaky bucket constrained at the entry to the switch, the crossbar architecture is capable of providing *exactly the same* delay guarantees as those possible in output-buffered switches. In particular, since it is possible to emulate, in principle, an output-buffered switch with a WFQ-like scheduler at the output, it means that it is theoretically possible to provide tighter delay guarantees than those obtained in this dissertation. However, the complexity of the algorithms for exact emulation appears to be prohibitively high for real-time high speed implementations. Of course, the problem of providing delay guarantees considered in this dissertation is a simpler problem that the much more ambitious problem of exact emulation. Therefore, it is not surprising that substantially simpler solutions can be found for a simpler problem.

The complexity of implementation of the algorithms for exact emulation of an output-buffered switch can be conceptually separated into three parts. First is the complexity of the computation of the stable matching. In the straightforward version of the algorithms described in [7], the arbiter needs to consider cells from potentially all flows to compute a stable matching. However, this complexity can be reduced by grouping flows by virtual output queues.

A more serious source of implementation complexity of algorithms for exact emulation stems from the fact that each input in the real switch should be aware of the ordering of its cells in their respective *output* schedules of the *emulated* switch. This implies that each output needs to be immediately informed of all new arrivals which occur during the current cell time, compute the position of all the new arriving cells in the emulated (WFQ-like) scheduler, and then notify the inputs of the newly computed value. In particular this could mean that even for a single arrival per cell time per input (as is assumed in [7]), a single output may need to inform *all* inputs of the position of the newly arrived cells in its schedule. This information must be relayed to the inputs during a single matching phase. The associated control communication overhead represents a serious bottleneck which needs to be overcome in order to allow for high-speed implementations of the exact emulation of output-buffered switches. In contrast, the approach described in this dissertation does not suffer from the control communication bottleneck, since the amount of communication it requires is negligible.

Finally, the third source of complexity of exact emulation is related to *obtaining* and *maintaining* the information that needs to be communicated. In particular, it appears to require that the departure time of each cell in the emulated output-buffered switch is known. The situation is made even more difficult by the fact that in the case of an emulated output-buffered switch with a WFQ-like scheduler, a single arrival may change the absolute departure times of all cells already in the switch destined to the same output. This will cause the necessity not only to compute the departure time of the newly arrived cell, but also to change the departure times of potentially *all* cells in the crossbar switch destined to the output corresponding to the new cell. All this needs to be accomplished in a single cell slot. These issues have to be addressed in order to allow real-time high-speed implementations of algorithms for exact emulation of output-buffered switch employing a WFQ-like scheduler. To the best of my knowledge, the solution to this problem is currently unknown.

Another limitation with the approach in [7] is the assumption that at most one cell can arrive at an input channel in one cell slot. This assumption is quite natural in the case when there is a single input link per input channel (of the same speed as the channel). The assumption of a single arrival per channel is essential in the proofs given in [7]. However, in practice it is frequently desirable to multiplex several lower-speed links into a higher-speed channel. To accommodate such multiplexing, most commercial switch designs accommodate several links (line cards) per channel. In this case, even if admitted rates do not exceed the channel capacity, many cells can arrive at the same input in a cell time, violating the essential assumption in the proofs of the results of [7]. In contrast, the approach described in this dissertation does not rely on the assumption of a single arrival per cell time, and can therefore easily accommodate multiplexing of several links on a single channel.

7.3 Areas for Future Research

While the algorithms for providing guaranteed delay described in this dissertation are, to the best of my knowledge, the most implementable among the existing work, the complexity of computing a maximal matching at the arbiter remains quite substantial. The challenge in providing delay guarantees in crossbars has been reduced to the challenge of building sufficiently fast arbiters. Since the arbitration algorithms described here requires very little control communication, the processing speed of the arbiter is the main factor determining the speed at which these algorithms can be run. With processing speeds doubling every year compared to an approximately 10% yearly growth in memory speeds, shifting the bottleneck from memory speeds to processing capacity appears to be a significant achievement. Yet, there is a lot to be done in this respect. It may be possible to find efficient hardware implementations for the arbitration algorithms described here. This will eliminate the computational bottleneck and hence reduce the required processing speeds.

Further, it seems that the performance bounds obtained here are quite loose. It seems that there is much room for further work in determining tighter delay bounds. It also appears that there may exist algorithms of similar complexity and with similar delay bounds which can be achieved with smaller speedup values. In particular, it is conjectured that using WF^{2}Q directly in the arbiter instead of FRECF in the framework discussed in Chapter 5 may yield the same delay bounds with substantially smaller speedup value. More specifically, it is conjectured that speedup $S = 2$ is probably sufficient to provide the same delay guarantees in this case.

The focus of this dissertation has been providing QoS guarantees for unicast traffic. With the proportion of multicast traffic growing every year, finding solutions capable of supporting multicast traffic in scalable switch architectures is necessary. Providing QoS guarantees for multicast traffic in traditional crossbar switches is a challenging problem which remains unsolved. Extending the results of this dissertation to multicast traffic, as well as finding other means for solving this problem, is an important area for future work.

In general, the need for providing high-speed switch implementations capable of supporting heterogeneous traffic requirements while handling multi-gigabit and tera-bit speeds call for a search for yet simpler algorithms capable of providing QoS guarantees.

Bibliography

[1] T.Anderson, S.Owicki, J.Saxe, C.Thacker, "High Speed Switch Scheduling for Local Area Networks", Proc. Fifth International Conference on Architectural Support for Programming Languages and Operating Systems", October 1992, Boston, MA.

[2] J.C.R.Bennett, H.Zhang, "WF^2Q: Worst-case Fair Weighted Fair Queueing", Proc. IEEE INFOCOM'96, March 1996, San Francisco, CA.

[3] J.C.R. Bennett, H.Zhang, "Hierarchical Packet Fair Queueing Algorithms", ACM SIGCOMM'96.

[4] J.C.R Bennett, Donpaul Stephens, Hui Zhang, "High Speed Scaleable and Accurate Implementation of Fair Queueing Algorithms in ATM Networks", ICNP'97

[5] D. Bertsekas, R.Gallager, "Data Networks", Prentice Hall, 1992.

[6] C.-Y Chang, A.J. Paulraj, T.Kailath, "A Broadband Packet Swith Architecture with Input and Output Queueing", Proc. Globecom'94

[7] S.T. Chuang, S.T., A. Goel, N. McKeown, B. Prabhakar, "Matching Output Queuing with a Combined Input Output Queued Switch", submitted for publication.

[8] L.Georgiadis, R.Guerin, V.Peris, "Efficient Network QoS Provisioning Based on per Node Traffic Shaping", Proc. IEEE INFOCOM'96.

[9] S. Golestani, "A Self-Clocked Fair Queueing Scheme for Broadband Applications", Proc. IEEE INFOCOM'94, June 1994.

[10] P.Goyal, H.M. Vin, H. Chen. "Start-time Fair Queueing: A Scheduling Algorithm for Integrated Services Packet Switching Networks", ACM SIGCOMM'96.

[11] R. Guerin and K.N.Sivarajan, "Delay and Throughput Performance of Speeded-Up Input-Queuing Packet Switches, "IBM Research Report RC 20892, June 1997

[12] A. Huang, G. Keisidis, "Guaranteed-Rate Round-Robin (GRRR) Scheduling for Input-Buffered ATM Switches", E&CE Technical Report no. 97-02, 1997

[13] I.Iliadis,W.Denzel,"Performance of packet switches with input and output queueing", Proc.ICC′90,1990.

[14] M. Karol, M. Hluchyi, S. Morgan, "Input Versus Output Queueing on a Space Division Switch," IEEE Transactions on Communications, vol. 35, 1987

[15] S.-Q. Li, "Performance of a Non-blocking Space-Division Packet Switch with Correlated Input Traffic", Proc. IEEE Globecom, 1989

[16] N. McKeown, V. Anantharam, J. Walrand, "Achieving 100% Throughput in an Input-Queued Switch", Proc.IEEE Infocom, 1996

[17] N. McKeown,"Scheduling Algorithms for Input-Queued Switches", Ph.D. dissertation,University of California at Berkley, 1995

[18] N. McKeown, M. Izzard, A. Mekkittikul, B.Ellersick, M.Horrowitz, "The Tiny Tera: A packet switch core", IEEE Micro, January 1997

[19] P.Krishna,N.Patel, A.Charny, R. Simcoe. "On the Speedup Required for Work-Conserving Crossbar Switches", Proc. IWQOS'98

[20] A.Parekh, "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks", Massachusetts Institute of Technology, Ph.D. Dissertation, June 1994.

[21] B. Prabhakar, N.McKeown, "On the Speedup Required for Combined Input and Output Queued Switching", Submitted to INFOCOM'99

[22] D.Stephens, "Implementing Distributed Packet Fair Queueing in a Scalable Switch Architecture", M.S. Thesis, Carnegie-Melon University, May 1998

[23] T. Rodeheffer, J.Saxe, Digital Equipment Corporation, personal communication.

[24] D. Stiliadis, A. Varma, "Providing Bandwidth Guarantees in an Input-buffered Switch", Proc. IEEE Infocom, 1995

[25] D. Stiliadis, A. Varma, "Frame-based Fair Queuieng: A New Traffic Scheduling Algorithm for Packet-Switch Networks." Proc. IEEE INFOCOM'96.

[26] I. Stoica, H.Zhang, "Exact Emulation of an Output Queueing Switch by a Combined Input Output Queueing Switch", Proc. IWQOS'98.

[27] J.Turner, "New Directions in Communications (or Which Way to the Information Age)", IEEE communications Magazine, 24, 1986.

[28] L.Zhang, "A New Architecture for Packet Switched Network Protocols", Massachusetts Institute of Technology, Ph.D Dissertation, July 1989.

[29] Ascend Communications. GFR family of Switches. www.ascend.com

[30] Digital Equipment Corporation. GIGAswitch. www.networks.digital.com