Technical Report

On Buffered Clos Switches

Santosh Krishnan
Henning G. Schulzrinne

Department of Computer Science
Columbia University
{*sk,hgs*}@*cs.columbia.edu*

Nov 01, 2002

Abstract

There is a widespread interest in switching architectures that can scale in capacity with increasing interface transmission rates and higher port counts. Furthermore, packet switches that provide Quality of Service (QoS), such as bandwidth and delay guarantees, to the served user traffic are also highly desired. This report addresses the issue of constructing a high-capacity QoS-capable, multi-module switching node.

Output queued switches provide the best performance in terms of throughput as well as QoS but do not scale. Input queued switches, on the other hand, require complex arbitration procedures to achieve the same level of performance. We enumerate the design constraints in the construction of a packet switch and present several approaches to build a system composed of lower-capacity memory and space elements, and analyze their performance. Towards this goal, we establish a new taxonomy for a class of switches, which we call Buffered Clos switches, and present a formal framework for optimal packet switching performance, in terms of both throughput and QoS.

Within the taxonomy, we augment the existing combined input-output queueing (CIOQ) systems with Aggregation and Pipelining techniques. Furthermore, we present the design and analysis of a novel parallel packet switch architecture. For the items in the taxonomy, we present algorithms that provide optimal throughput and QoS, in accordance with the above performance framework. While some of the presented ideas are still in the investigative stage, we believe that the current state of the work, especially the formal treatment of switching, will be beneficial to the ongoing research in the field.

Contents

Chapter 1

Introduction

The past decade has witnessed a manifold increase in the demand for switches that employ packet-based forwarding. The networking protocols supported by these systems vary from connection-oriented to connectionless ones, including Asynchronous Transfer Mode (ATM), Multi-Protocol Label Switching (MPLS) [60], the ubiquitous Internet Protocol (IP), and switched Ethernet. Irrespective of the specific protocol supported, these switches consist of a *forwarding path* that transfers data units, called a *packet* in the generic sense, from the input to the output interfaces, optionally providing preferential treatment, or Quality of Service (QoS) guarantees, to specified user flows.

Widespread consumer and business adoption of the Internet, together with improvements in the physical layer technologies, has resulted in increased demand for packet switches with very large capacities, in terms of both the number of supported interfaces as well as the interface transmission rates. For example, a typical IP router in production today supports 10 to 20 ports, each at 10 Gb/s, requiring a *forwarding path capacity* in the vicinity of 200 Gb/s. Since packet sizes have not changed materially with the increasing rates, assuming a worst case of 64-byte packets, the router also requires a *packet processing capacity* of about 20 million packets per second. This represent at least a tenfold increase in capacities since just half a decade ago. Moreover, this trend is expected to continue with the anticipated adoption of 10-G Ethernet in the metropolitan area networks, and OC-192 (10 Gb/s) and OC-768 (40 Gb/s) links in the long haul. This motivates the design of switch architectures that can scale with increasing rates and port counts. In general, an increase in the desired forwarding path capacity imposes design constraints on the construction of the switch, while the packet processing capacity constrains the complexity of the algorithms that operate on a per-packet basis.

As an orthogonal trend, in an attempt to benefit from the efficiencies of a packet-based network, traditionally circuit-switched network operators are migrating towards the former in order to support customer *flows* that require service guarantees or QoS. Examples of such flows include Voice-over-IP (VoIP) sessions, packetized Video-on-Demand, and MPLS Virtual Private Network (VPN) tunnels with guaranteed bandwidth between office locations. The implication of this trend for switch design is the need to implement scheduling and buffer management techniques in the forwarding paths, so that the specified service requirements may be met.

This report addresses both the above issues, and presents architectures and algorithms that can be used to construct the forwarding path of a large-scale packet switch that also guarantees QoS to the supported flows. Towards this goal, we propose a new taxonomy for a class of scalable packet switches, together with an analytical framework to characterize different levels of optimal performance for the algorithms employed within such switches. At present, some of the results are at a conjectural stage, nonetheless, we believe that the current state of the work, especially the formal treatment of switching, will be beneficial to the ongoing research in switching.

1.1 Motivation

There is a vast amount of prior work in the design and analysis of circuit switches (see, for example, [29] for an overview, and Appendix A for a primer on basic switching concepts and commonly used terms). Circuit switching has benefitted immensely from a well established performance framework, namely, different levels of *blocking* behavior, which characterizes the performance of a given design in isolation from the properties of specific circuit arrival processes. All non-blocking circuit switches are *functionally equivalent* in terms of the circuits that they can admit. This has driven much of the work in the construction of scalable circuit switches using an inter-connection network of smaller components, examples of which include the Banyan, Benes, Clos and Cantor networks.

Traditional packet switching theory has borrowed, often erroneously, many of the terms, constructs, and performance measures from the former. However, packet switching is fundamentally different due to the presence of external link contention and buffers in the switch components. There exists no uniform framework that characterizes optimal performance, or which provides a notion of functional equivalence with an ideal switch. Consequently, several conflicting measures have been used to claim optimality. For example, the term blocking, which is really a circuit switching concept, has been variously used to mean either a blocking structure, or *head-of-line* blocking, or non-work conserving. Some other approaches use the notion of 100% throughput (100% compared to what?) to characterize optimal performance, while quite a few study a switch design by computing the loss ratios, for the finite buffers within the switch, for specific arrival processes (most likely an artifact of the arrivals themselves rather than that of the switch design). In summary, a framework which is able to encompass optimal throughput and QoS properties of a switch *design* will be beneficial towards the work in complex scalable packet switches.

The problem of scaling packet switches by using multiple stages of elements has been addressed in the past, and a taxonomy of ATM switch architectures can be found in [57]. However, most of the work deals with the construction of the inter-connection network within the switch, the focus being solely on the structural blocking property of the network. Further studies (e.g., [7]) have characterized the throughput performance of a specific class of Banyan-based topologies. Though such works have enabled to implement practical multi-stage switches, providing functional equivalence with an ideal switch remains an open topic of research for many of the simple non-blocking networks augmented with buffers at arbitrary points within them. This report takes a small step in that direction.

1.2 Our Contributions

We address the problem of constructing a high capacity packet switch, using stages of lower capacity memory elements and non-blocking space elements. We restrict ourselves to designs that are structurally non-blocking and resemble the three-stage Clos network, with the new performance framework of functional equivalence. For switches with this structure, which we call *Buffered*

Clos switches, we establish a *taxonomy* of designs driven not by the switch structure alone but by the methodologies employed to overcome various design constraints. Each item in the taxonomy is accompanied by the constraint it resolves, so that switch implementors may identify the most appropriate one to use. While some items in the classification, such as the combined input-output queueing (CIOQ) switch, are already addressed to a large extent in the recent literature, the performance framework and the taxonomy itself are novel. In addition to providing a road-map to construct scalable switches, it also allows to ascertain the properties of existing vendor equipment by inspecting the structure and the associated algorithms for functional equivalence.

Within the above taxonomy, we also present the design and analysis of CIOQ switches with *aggregation* and *pipelining*, two crucial ingredients to scaling. These fall into the category of single-path buffered Clos switches. Some existing switches appear similar to these, nevertheless, the presented algorithms and analytical treatment are novel. Furthermore, we propose and analyze a new *parallel* packet switch architecture, which belongs to the multi-path category. Finally, we propose a new methodology to enable QoS guarantees by combining certain levels of function equivalence with existing scheduling schemes.

1.2.1 Applicability and Scope

This work is restricted mainly to the treatment of unicast traffic flows. While the ability to handle multicast traffic is certainly relevant to packet switching, we do not deal with it in any level of detail in order to keep this work tractable. For the same reason, we also do not include specialized node algorithms that may be used to optimize the performance of adaptive traffic. Our assumption is that the optimal throughput property by itself is beneficial to adaptive traffic, and any other mechanisms, such as active queue management (AQM) to enforce fairness among adaptive flows, can be added on to this work without much difficulty. In addition, we do not address the end-to-end network behavior, and network engineering. Instead, we concentrate on node mechanisms whose suitability for end-to-end behavior may be studied independently. Finally, this work is agnostic of the protocol processing specifics and control plane handling, e.g., signaling the QoS flows, and route control.

1.3 Outline of the Report

The remainder of this report is organized as follows. Chapter 2 introduces the switching model including the components used and meaningful notions of optimal performance. Chapter 3 covers the related work in switch design and provisions for QoS. Chapter 4 contains our original contributions, including the taxonomy of Buffered Clos switches and the framework of functional equivalence. It also includes the completed work on aggregated and pipelined CIOQ switches, the parallel packet switch architecture, and our proposed QoS methodology. Finally Chapter 5 concludes this report with an identification of related future work.

Chapter 2

Switching Framework

We now present the switching framework, including some meaningful notions of optimal performance. We then overview the models for the forwarding path of a packet switch, and their atomic components. Then, we briefly introduce the output queued and input queued switching models, the former representing the ideal reference switch for equivalence purposes.

2.1 Notions of Optimal Performance

A circuit request between an input-output link pair, in the context of circuit switching, is considered *admissible* if there is capacity available on the specified links. A non-blocking switch ensures that a path within the switch is realizable for every admissible circuit request. Since circuit admissibility on the external links constrain the maximum possible circuit throughput, we may assert the following:

Proposition 1 *The non-blocking property is necessary and sufficient to ensure optimal throughput of a circuit switched node.*

Similarly, in the context of packet switching, a flow between an input-output link pair with a given effective bandwidth is considered admissible if the bandwidth is available on the specified links. The successful admission of this flow depends solely on the existence of a switching path, with the said bandwidth (and associated buffers) available throughout the path. In other words, if the flow bandwidth can be accommodated on the input and output links of the switch, this existence ensures that the same can be accommodated throughput the switching path. By definition, since the external links impose a physical limit on the total amount of flow bandwidth that can be admitted into the switch and consequently any notion of an optimal set of flows, by recognizing the analogy with circuit switched nodes, we may assert the following (without analytical proof):

Proposition 2 *A packet switch design which is* structurally equivalent *to a non-blocking circuit switch is necessary and sufficient to admit an optimal set of flows with bandwidth requirements.*

The sufficiency assumes that the scheduling and buffer management schemes can guarantee the allocations, itself a non-trivial task. We will refer to such switches as *structurally non-blocking* packet switches, and the act of allocating bandwidth and buffer resources throughout the switching path as *flow fitting*. Since much of the early work was based on ATM switches, wherein the entire traffic undergoes CAC and all the flow specifications are known prior to actual packet transmission, the ability to fit (and schedule) flows was the primary objective, leading to vendor claims of optimal performance based only on the non-blocking property. Even though this may constitute the first step in packet switch design by enabling flow fitting, ensuring a non-blocking structure cannot be equated to providing optimal throughput.

Limited by the physical constraints of the input and output links, maximum packet-switched throughput may be achieved as long as an output link of the switch never idles when a packet destined to it is backlogged anywhere within the switch. Switches with this property are referred to as *work conserving* and a given switch design is considered *ideal* (in throughput) if it satisfies this property. If it is difficult or impossible to prove this property, one may justifiably claim optimality by proving a *functional equivalence* with a well known ideal switch. Similar to the case of circuit switch design, where the non-blocking property ensured functional equivalence with a well known optimal switch (e.g., a single-stage crossbar), an unambiguous framework is required to characterize functional equivalence of packet switch designs.

Proposition 3 *A packet switch which is* functionally equivalent *to a well known ideal switch is optimal in throughput.*

We can identify a few approaches to functional equivalence, which address the construction of a switch rather than the specifics of the arrival traffic procceses. The first relies on constructing a switching fabric S_1 in such a manner that the packets depart from the outputs in an *exact emulation* of the departure from a well known ideal switch S_2 of the same dimensions, for *any* traffic pattern. The second approach relaxes the constraint of exact emulation and instead borrows from the concept of stability of queueing systems [55]. A switch with a set of queues is said to be *absolutely stable* if the expected queue occupancies are finite. Clearly, absolute stability depends on the characteristics of the arrival process in addition to the design of the switch. Therefore, we refer to a fabric S_1 as *relatively stable* with respect to S_2 if it is stable for a broad class of arrival processes for which the latter is also stable. We may refer to S_1 as relatively stable in the *strict sense* if the subset of queues that are stable in the reference switch remain so in S_1, under any arrival pattern. The strict sense property allows to consider traffic patterns that can lead to (partial) instability in the ideal switch, and to handle it in the same manner as the latter. The final approach further relaxes the stability constraint by relying on a prior knowledge of flow rates. If the rates of all the flows (the actual offered rates, *not* the rate requirements) traversing the switch can be pre-computed, optimal throughput is achieved just by the ability to fit flows, i.e., the structural non-blocking property.

To summarize, the structural non-blocking property, along with the appropriate scheduling and buffer management policies, provides the framework for QoS in a packet switch, and a functional equivalence with an ideal switch provides the framework for throughput performance. Optimal packet switching performance may be achieved by the simultaneous satisfaction of both of the above.

2.2 Forwarding Models

The first generation routers, and many of the current lower speed ones, perform software-based packet forwarding. A general purpose CPU is connected to multiple line cards through an I/O bus, as shown in Fig. 2.1(a). The cards perform the physical and link layer protocol processing and forward the incoming packets to the CPU, where the packet header is analyzed to determine

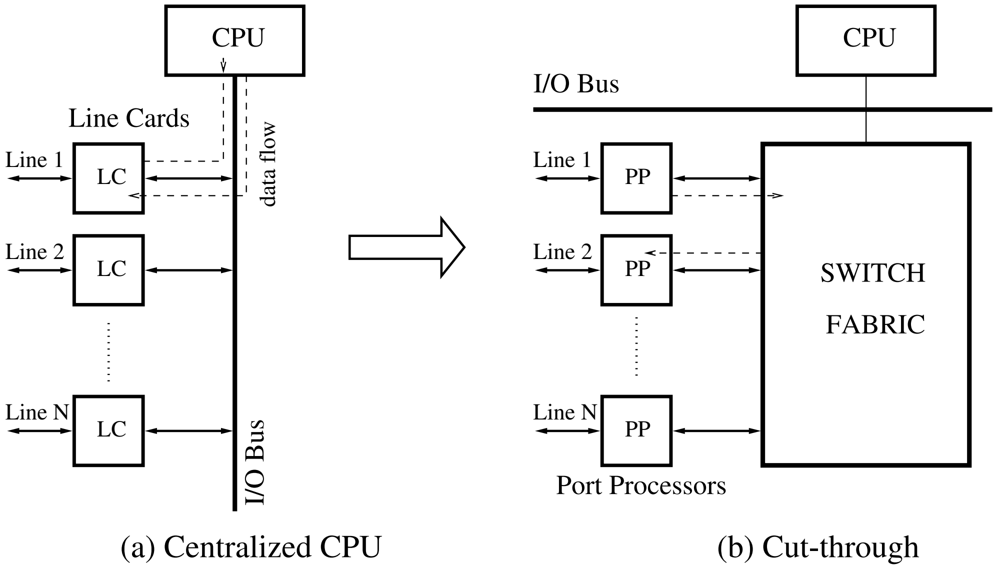
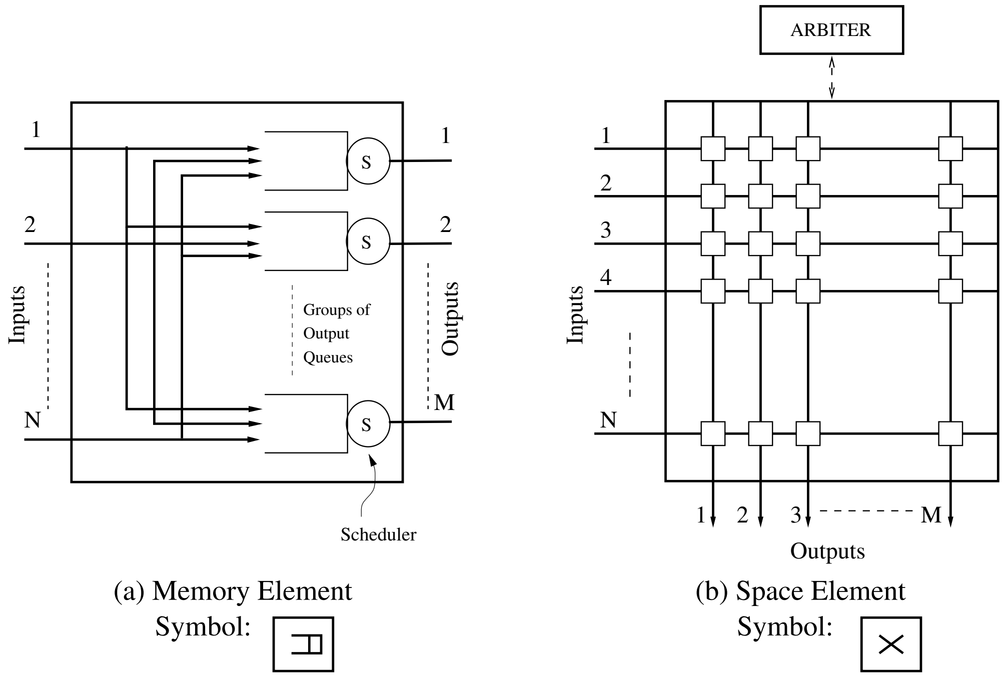


Figure 2.1: Forwarding models: (a) Centralized CPU, (b) Cut-through

the outgoing interface and any special handling, if applicable. Buffers are distributed among the line cards and the CPU memory, to account for the output, I/O bus and CPU contention. The disadvantage of this model is that the limited communication bandwidth of the I/O bus and the software module on the CPU present significant bottlenecks to packet throughput, and thereby limit the port counts and the interface rates that can be supported. For example, a 16×16 switch, with 1 Gb/s ports, requires (i) a processing capability of 32 million (64-byte) packets per second, which translates to an upper bound of 32 (instructions plus memory) cycles of a 1 GHz CPU; (ii) an I/O bandwidth of 32 Gb/s, compared to the 8 Gb/s available using the latest 64-bit 133 MHz PCI bus; and (iii) a memory bandwidth of 32 Gb/s, allowing merely 16 ns for a memory transfer even assuming a total bus width of 64 bytes. Clearly, these are tall orders even for the moderately sized switch of the example. Consequently, the path traversed by the packet through the central CPU is referred to as the *slow path*.

Modern designs use the *cut-through* model illustrated in Fig. 2.1(b). The incoming packets undergo header processing in the port processor cards attached to each interface, and are directly routed through the switch fabric, or the *fast path*, to the corresponding output interfaces. The central CPU is connected via a special switch port. Only the control traffic is routed to the CPU, which maintains the overall state of the switch and programs the cards. In addition to the physical and link layer processing, the components on the cards are responsible for address lookup, flow filtering, policing, statistics collection and other protocol specific tasks, and for appending the results of those operations into a special local header. For the above example, this model requires a processing capability of 2 million packets per second in each card, a switching fabric aggregate capacity of 16 Gb/s, and a memory bandwidth that depends on the switch design, but no more than 32 Gb/s. Communication IC vendors already offer network processors that implement those components in silicon at interface rates of 10 Gb/s, processing capabilities of 20-30 million packets per second, and switching fabrics that support up to 160 Gb/s.



(a) Memory Element
Symbol:

(b) Space Element
Symbol:

Figure 2.2: Forwarding elements: (a) memory element, (b) space element

2.3 Basic Switching Elements: Building Blocks

A fabric may be constructed using one or more instances of basic forwarding elements, of two types: the *memory element* and the *space element*. An $N \times M$ memory element, shown in Fig. 2.2(a), accepts packets from the N input interfaces and enqueues them into one or more (in the case of multicast) queues, which are arranged into groups, one for each of the M outputs. Note that the queues need not be physically separate in memory, and may indeed share packet buffers from a common pool. The structure of a group may be as simple as a single output queue per interface, or may allow grouping packets on the basis of several criteria such as input interface, flow identity, and priority, in order to facilitate preferential treatment. A buffer management logic is responsible for admitting packets into the queues, while a scheduler logic at each output interface dictates the outgoing packet rate and the relative order of the packets. The latter may vary from a first-in first-out (FIFO) policy acting on a single output queue, or a sophisticated weighted fair queueing (WFQ) policy acting on per-flow queues. Given an interface rate of C and a minimum packet size of L, the element needs a memory bandwidth of $(N + 1)C$ at each of the outputs to accommodate N simultaneous enqueue operations (from the N inputs) and a single dequeue operation. At the same time, the scheduler logic needs to dequeue at a frequency of C/L to sustain the output interface rate. Hence, the dimensions and the rates that can be supported by a single memory element is limited by the available memory bandwidth, while the interface rates along with the smallest packet size limits the complexity of the scheduler. Memory elements available today operate in the order of tens of Gb/s.

A space element, shown in Fig. 2.2(b), is composed of a bufferless cross-connect, and an arbiter (possibly distributed) that is responsible for configuring the inputs and outputs into a matching. Packets on the input interfaces are segmented into equal sized *cells*, if necessary, and in each synchronous time unit of the element, called a *timeslot*, a single cell is transfered from the inputs to the outputs, configured by the matching. The segmentation and re-assembly of packets is not the responsibility of the element, and is typically implemented in one of the components of the line cards. The matching is re-computed at every timeslot, and therefore the arbitration logic needs to operate at a frequency of C/L, while the complexity of the logic is determined by the size of the element. While it is the frequency and complexity of arbitration that eventually limits the dimensions of a single element, we should note that the chip size and the underlying technology also affects the interface counts and rates that may be supported. Electronic crossbars available today are capable of interface rates in the order of 1-10 Gb/s, with port counts in the order of 20-30, and can be re-configured in less than 100 ns thereby providing a very small timeslot. Optical components such as wavelength routers [71] can support a significantly higher interface rate, limited only by the optical physical layer. However, such components are expensive, the port counts supported by the currently available ones are lower, and most significantly, the re-configuration times are in the order of a few μs, thus providing a very coarse-grain timeslot.

Note that we consider the basic elements as *logical* entities, i.e., as black boxes. The implementation of a memory element subsumes the existing work on different internal logic that may be employed to exhibit the behavior described above. Similarly, the techniques from circuit switching may be used to construct a large non-blocking inter-connection network that emulates a single $N \times M$ logical element.

2.4 Common Existing Solutions

A packet switch that employs the cut-through forwarding model, with a single memory element for the switch fabric is referred to as an *output queued* (OQ) switch. Several of the early switches used this design, an overview of which can be found in [69], in which a few alternative approaches were used to design the $(N + 1)C$ capacity queueing logic. For example, the Prelude switch [21] uses a shared timeslot exchanger to realize output queueing, whereas the knockout switch [72] uses N parallel shared buses for the same purpose. While the implementation of the queueing logic is beyond our consideration, it suffices to note that pure output queued designs are rarely used in current high speed switches due to the significant memory bandwidth bottleneck. Output queueing may also be emulated by physically grouping queues on the basis of input-output pairs, and either providing $N \cdot M$ disjoint paths between the N inputs and M outputs to form a bus-matrix switch, or by placing those queues at the crosspoints of an $N \times M$ crossbar. While both these approaches reduce the memory bandwidth requirement to $2C$, the chip size imposes a restriction on its dimensions.

Assuming that the port processors and the queueing logic operate at (interface) wire speed, an output queued switch does not suffer from any internal contention, and hence has the ability to be work conserving. Furthermore, since packets are directly presented to the output queues without any intermediate delay, the flow rates and packet delays can be tightly, controlled by the output scheduler. Consequently, given the required interface counts and rates, a hypothetical output queued switch of those dimensions may be considered an ideal reference switch for performance comparisons.

In contrast to the above, an *input queued* (IQ) switch concentrates all the packet buffers at the input interfaces of the system. The switch fabric consists of a single logical $N \times M$ space element, with N (1×1) memory elements connecting each of the input interfaces to the input ports of the space element. The latter may be collocated within the port processing cards of the switch. Notice that such a design suffers from internal contention at the inputs, i.e., the memory elements do not function in a work conserving manner and are driven by the sequence of matchings computed by the arbiter of the space element. Therefore, while the maximum memory bandwidth required for an input queued switch is $2C$ irrespective of port count, complex arbitration policies are required to achieve functional equivalence with a reference ideal switch. This has been one of the primary subjects of packet switching research in the recent past, and notable results will be reviewed in the next chapter.

Chapter 3

Related Work

We now review specific parts of the literature that relate to this work. First, we show how to construct a Clos network, in the context of circuit switching, and discuss its applicability to packet switches. Next, we outline some of the recent results in IQ and CIOQ switches, which form the starting point of our taxonomy. Finally, we review the work on QoS methodologies for packet switches.

3.1 Clos Network

The Clos network is the simplest three-stage non-blocking arrangement, composed of space elements in each stage. While it was first introduced and analyzed in the 1950s, it remains relevant to this date, and we borrow this overview from Pattavina's more recent work [57]. As shown in Fig. 3.1, an $N \times N$ switch is composed of three fully connected stages, where each link has the same capacity. The first stage is composed of M instances of $N/M \times K$ non-blocking space elements. These are inter-connected to the third stage composed of M instances of $K \times N/M$ elements, using K elements of size $M \times M$ in the central stage. The network is non-blocking in the strict sense if and only if $K \geq (2N/M - 1)$ (Clos theorem). The speedup required between the first and second stages can be easily calculated as $(2 - M/N)$. Provided the above condition is satisfied, given an idle input-output pair, a greedy algorithm which visits each of the central element is guaranteed to find a path in $O(N/M)$ time.

The network is re-arrangeably non-blocking if and only if $K \geq N/M$ (Slepian-Duguid theorem), and the re-arrangements necessary to realize a path for an idle pair can be found in $O(M^2)$ time (a consequence of Paull theorem). The minimum number of crosspoints in the network is $O(N^{1.5})$ which results when $M = \sqrt{2N}$. Notice that no speedup is required if re-arrangement is allowed. The results continue to hold when we aggregate the links (in and out) of a space element to a single faster physical link. Specifically, this implies that N/M simultaneous matchings on the (aggregated) input-output pairs can be partitioned into K concurrent matchings on the (pipelined) central stage. In other words, the fan-in and fan-out of the space elements may interchangeably be in space or time. Strict-sense non-blocking refers to the case when the existing partitions may not be touched to admit a new entry, while a re-arrangement refers to the ability to re-compute the partitions on every circuit admission.

An interesting aspect of the above properties is that similar results can be obtained when the quantities being fitted on the links are any additive scalars. Let the capacity of each link in Fig. 3.1

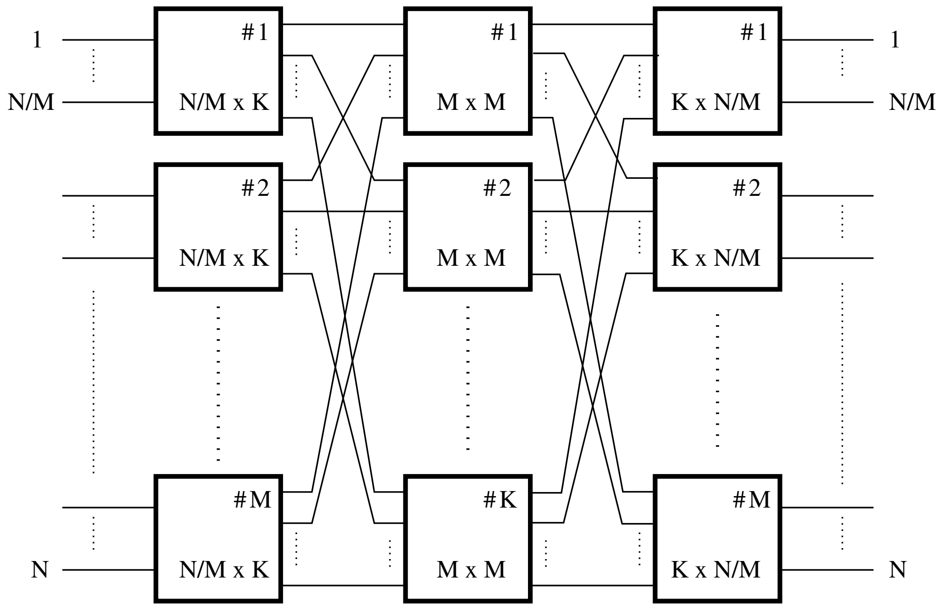


Figure 3.1: An $N \times N$ Clos Network

be normalized to 1 unit, and let the quantities to be fitted on an input-output pair vary within $[b_{\min}, b_{\max}]$. A quantity b is admissible for the pair (i, j) if both input i and output j have current allocations that do not exceed $(1 - b)$. We can prove that a path can be realized for an admissible quantity as long as the following condition holds [54]:

$$K \geq 2 \left\lfloor \frac{(N/M) - b_{\max}}{1 - b_{\max} + b_{\min}} \right\rfloor + 1$$

This reduces to the Clos theorem when all quantities are unity, the extra terms accounting for fragmentation of the link capacities due to the variability in the quantities. Note that the scalar quantities may represent the peak or the effective bandwidths of packet flows. Hence, we may construct a three stage network that is structurally equivalent to the Clos network, and enable fitting of admissible flows, irrespective of whether the nodes are memory or space elements, as long as each element can internally (through arbitration or scheduling) realize the bandwidths of the individual flows on their input and output links. This result mirrors Proposition 2 in the context of Clos topologies.

3.2 IQ Switch Algorithms

Since IQ switches require a memory bandwidth that is independent of the switch size, a significant amount of research has been conducted to characterize its performance. Consider an $N \times N$ IQ switch, as defined in Sec. 2.4, with all rates normalized to an external interface rate of 1 cell/timeslot, and mean arrival rates of $\lambda_{i,j}$ between input i and output j. These rates are said to be *admissible* when $\sum_i \lambda_{i,j} < 1$ for all j, and $\sum_j \lambda_{i,j} < 1$ for all i, since they correspond to arrivals for which an OQ switch of the same dimensions is stable. The task is to devise arbitration

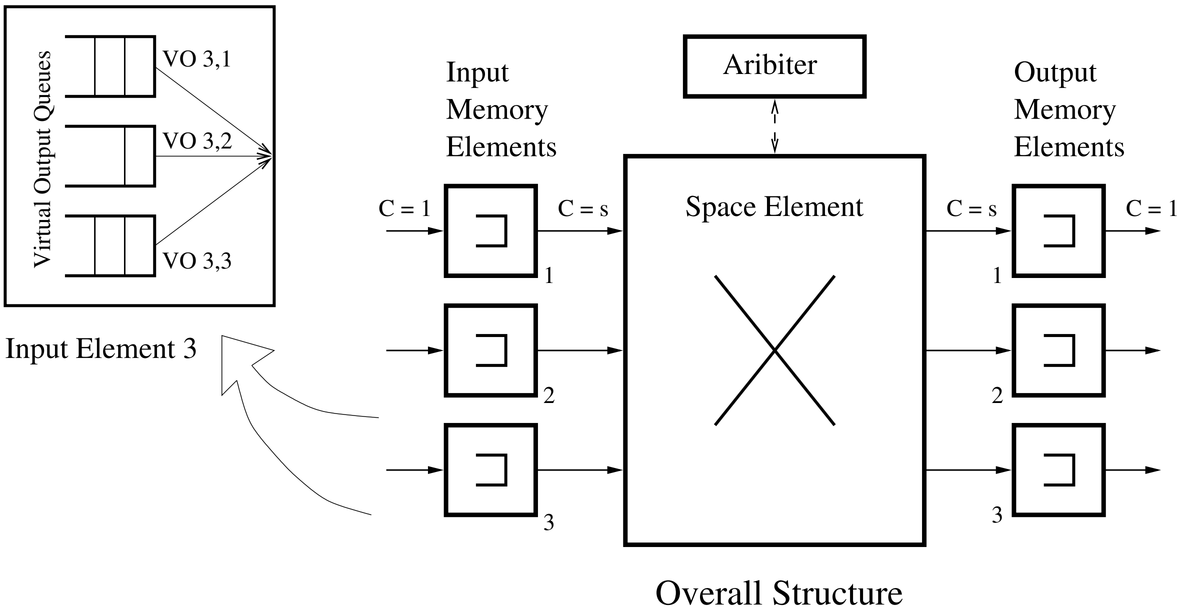


Figure 3.2: A combined input-output queueing switch (CIOQ)

algorithms[1] that compute a sequence of matchings $\pi(n)$, so that the observed behavior is functionally equivalent to the OQ switch. Broadly speaking, there are two types of such algorithms: *rate driven* schemes, which use a prior knowledge of all the rates $\lambda_{i,j}$, and *queue-occupancy driven* schemes which use the state of the queues to devise a matching. The latter is especially suitable when the rates are not known beforehand, as is the case with best-effort traffic, or when an equivalent rate-driven scheme is too complex to implement.

Early IQ switches used a single FIFO queue in each of the input memory elements. However, Karol *et al.* [37] showed that the throughput of this system is limited to $2 - \sqrt{2}$ (59%), even under independent Bernoulli traffic on each input link, which is uniformly distributed to all the outputs. This is because of the phenomenon called head-of-line (HOL) blocking, in which a cell destined to an idle output is queued behind a cell destined for a currently busy output. This situation is remedied by queueing the cells in an input element into N separate queues, one for each output, referred to as *virtual output queues* [68]. Let the queue of cells destined to output j in input i be denoted as $Q_{i,j}$, with length $q_{i,j}$. Essentially, a matching matrix $\pi(n)$ enables the transfer of a cell from a non-empty $Q_{i,j}$ in timeslot n whenever $\pi_{i,j}(n)$ is set to 1.

The *size* of a matching is the number of the unity elements in the matrix, i.e., $\sum_{i,j} \pi_{i,j}(n)$, and corresponds to the total number of cells transfered in the timeslot. Given a *weight* function, the weight of a matching is defined as $\sum_{i,j} w_{i,j}(n) \cdot \pi_{i,j}(n)$. An IQ switch is said to employ a speedup $s > 1$ if s matchings are computed per timeslot, enabling the transfer of up to s cells per timeslot between an input-output pair. Since the output link operates at a rate of 1 cell/timeslot, any speedup necessitates buffers at the front of the output link, in addition to the ones in the input elements. Such a switch is also referred to as a combined input-output queueing (CIOQ) [31] switch and is illustrated in Fig. 3.2. Note that a CIOQ switch requires a memory bandwidth equal to $1 + s$ times the interface rates, and a CIOQ switch with speedup N is the same as an OQ switch.

[1]Such algorithms have also been referred to as switch *scheduling* algorithms, but to avoid confusion with the output link scheduling algorithms, we refer to these as arbitration algorithms.

3.2.1 Size Matchings

An algorithm that computes a matching with the maximum size would provide the highest *instantaneous* throughput in each timeslot. Such a matching can be obtained by finding the maximum network flow in a bipartite graph, where the vertices of the graph correspond to the inputs in one set and the outputs in the other, and an edge with unit capacity exists between input i and output j if $q_{i,j} > 0$. The most efficient algorithm to solve this problem runs in $O(N^{2.5})$ time [28], and is too complex to implement at a high frequency (e.g., 20 million arbitrations/second for an interface rate of 10 Gb/s and cell size of 64B, allowing less than 10 cycles on a 133 MHz clock for the arbitration logic). This motivated the design of practical so-called *maximal*-size [50] matching algorithms.

A matching is referred to as maximal in size if it adds one connection at a time, i.e., in a greedy approach, and converges when there is no idle input-output pair. Clearly, any maximal matching algorithm will converge in N steps, though it is not guaranteed to find the maximum size match. A simple example may be constructed as follows: a centralized arbiter matches each input i, in sequence, to the first unmatched output j for which $q_{i,j} > 0$. This algorithm runs in $O(N^2)$ time, which is the minimum it would take for any algorithm that needs to sequentially inspect all the entries of an $N \times N$ matrix. Dai and Prabhakar [19] proved, using fluid analysis, that *any* maximal matching algorithm guarantees that the system of queues is stable for *any* admissible arrival process, with a speedup $s \geq 2$. This is a remarkable result as it allows one to choose the least complex maximal matching, and yet remain stable. Similar results, i.e., the sufficiency of $s = 2$, were shown by Leonardi *et al.* [48] for a specific maximal matching under admissible traffic which leads to an embedded Markov-chain queue length evolution, by establishing a drift condition on a well chosen quadratic Lyapunov function [46] of the queue length vector. The same work [48] also established the stability of a probabilistic (queue-length driven) maximal matching.

Several earlier works [1, 51, 49] proposed matching algorithms for IQ switches without speedup, and analyzed their performance under restricted arrival processes, as well as sub-maximal sizes. In Parallel Iterative Matching (PIM) [1], each input, in parallel, sends a request to each of the outputs for which $q_{i,j} > 0$. An output chooses one from the contending inputs in a random manner, and finally, each of the inputs select one from the granting outputs, again in a random manner. These steps are iterated till the algorithm converges to a maximal matching. Even though N iterations are required to converge in the worst-case, it was shown that the algorithm needs only $O(\log N)$ iterations on average. Alternatively, in iSLIP [51], in each iteration, the output grant and the input select operations are performed in a round-robin fashion, with special care to ensure that the pointers do not synchronize. It was shown that due to the de-synchronization property, a *single* iteration is sufficient to ensure stability for i.i.d. Bernoulli traffic on each input, with uniformly distributed destinations. For uniform bursty traffic, several iterations, N in the worst case, are required to maintain stability. Note that such iterative algorithms, due to their parallel nature, can conceivably be implemented in a distributed fashion, as long as there is enough control bandwidth between the inputs and outputs to exchange results repeatedly (equal to the number of iterations) within the same timeslot.

3.2.2 Weight and Rate Matchings

Even though the performance of maximal size matching algorithms, without speedup, have been demonstrated for uniform traffic, it was recognized [52] that even a *maximum* size matching is insufficient for stability under non-uniform admissible traffic. The same work showed that a maximum *weight* matching, where the weight of the flow from i to j is the queue length $q_{i,j}$, is sufficient to ensure stability for any admissible i.i.d. stationary arrival pattern. The i.i.d. and stationarity requirements, which were the artifacts of the proof methodology that used embedded Markov chains, were relaxed in [19]. The most efficient algorithm to compute the maximum weight runs in $O(N^3 \log N)$ time. It was later realized [53] that the *best* maximum size matching can indeed guarantee stability by computing all possible maximum matchings, in each timeslot, and choosing the one with the maximum weight, where the weight $w_{i,j}$ is computed as the sum of all the occupancies in input i plus the sum of occupancies in output j.

Given a prior knowledge of all the rates $\lambda_{i,j}$, either through traffic engineering considerations, or through specifications of traffic profiles, a repeating sequence of K matchings may be used to essentially provide a virtual trunk with the specific rates, thus ensuring stability. The K matchings may be computed offline using the Birkhoff-Von Neumann (BVN) matrix decomposition [5], which computes $K = O(N^2)$ number of templates in a total run time of $O(N^{4.5})$ and requires no speedup to operate. Note that while the algorithm has high complexity and memory requirement to store the templates, the decomposition needs to be repeated only when the rate matrix changes. Alternatively, the templates can also be computed by performing flow fitting on a structurally equivalent Clos network [30]. (This may be viewed as a proof for Proposition 2 for the specific cases of IQ and CIOQ switches.) In this approach, assuming that all the rates are integer multiples of some $\lambda_{\min}$, each rate is converted into $\lambda_{i,j}/\lambda_{\min}$ circuits, which need to be realized in $1/\lambda_{\min}$ time. This can be done with a speedup of 2, i.e., by partitioning the circuits into $2/\lambda_{\min}$ templates using the Clos theorem, or without speedup using the Slepian-Duguid theorem. The running time complexity for the former can be found as $O(N/\lambda_{\min}^2)$, and for the latter as $O(N^2/\lambda_{\min})$. In contrast to deterministically computing and storing templates, a probabilistic rate-driven algorithm was presented in [48], where the rates were used to generate a maximal size matching.

To summarize, a maximal size matching is sufficient to ensure stability under any admissible traffic using a speedup of 2, while a maximum weight matching ensures the same without speedup[2]. A maximal size algorithm, such as iSLIP, may be used without speedup for uniform traffic, and requires only a single iteration for uniform Bernoulli traffic. Finally, if all the rates are known beforehand, an offline rate decomposition algorithm may be used to generate a sequence of templates to provision those rates.

3.2.3 OQ Emulation

While different size, weight and rate-based matchings have been shown to possess varying stability properties, the ability to *exactly* emulate an OQ switch for any traffic pattern would undoubtedly be the most desirable property for an input queued switch. Chaung *et al.* [17] showed that this was indeed possible, in theory, for a CIOQ switch with a speedup of 2. (This was also proved independently in [44, 66].)

The algorithm presented in [17], Critical Cells First (CCF), maintains a priority list, for every output, of cells enqueued anywhere in the system. The position of a cell in this list, called its output *cushion*, is calculated by simulating an OQ switch of the same dimensions with the desired link scheduling scheme. The class of scheduling schemes which can be emulated is the Packet-In

[2]It has been widely conjectured that these two results are analogous to the Clos theorem and the Slepian-Duguid theorem, respectively, for a circuit-switched Clos network, as a matching can be viewed as an online fitting of circuits over the time-scale of averaging of the rates.

First-Out (PIFO), which is a model for any service discipline in which the relative departure order of cells does not change with future arrivals. An incoming cell is inserted into the input queues in such a way that the cells are in ascending order of their output cushions. A stable matching is calculated which ensures that, for each input, either a cell is transfered or a cell from another input with a lower output cushion is transfered. Such a matching can be computed by solving the stable marriage problem, for example, using the Gale-Shapely algorithm which runs in $O(B)$ time, where B is the sum of the lengths of all the input queues.

3.3 QoS Methodologies

As opposed to IQ arbitration algorithms, the main focus of which (except for rate decomposition and emulation schemes) has been to provide maximum throughput in relation to a work conserving ideal switch, the primary goals of QoS algorithms are twofold: (i) to provide *isolated* bandwidth guarantees to specified flows; and (ii) to provide fairness in the distribution of excess (best-effort portion) bandwidth. Isolation refers to the ability to provide a virtual bandwidth trunk to a flow, irrespective of the arrival patterns of other flows. The time-scale over which such isolation is provided translates into the *latency* of service, and consequently affects the delays observed by the packets belonging to that flow. Though several notions of fairness may be contemplated depending upon the application of the flow (such as TCP connections), we restrict ourselves to the ability to control the excess allocation depending on specified weights. Both the above objectives are especially meaningful under inadmissible traffic.

The goals of isolation and fairness are provided by a combination of scheduling and buffer management algorithms. These have been studied extensively in the context of OQ switches, i.e, single memory elements, and are overviewed in Appendix B.

3.3.1 QoS in Input Queued Switches

While OQ emulation established the theoretical possibility to provide any desired QoS performance in a CIOQ switch by emulating a chosen link scheduler for an equivalent OQ switch, its implementation remains infeasible. Accordingly, in practice, QoS is provided by considering each of the stages as independent elements and using a hierarchical scheduling structure, implemented in a distributed fashion [63, 8] among the elements.

We will consider a simple example shown in Fig. 3.3 to illustrate the methodology. The input elements contain a virtual output scheduler (VOS) for each output, which are served by the arbiter. Each VOS contains a GBS (guaranteed bandwidth scheduler) and an EBS (excess bandwidth scheduler) portion, which divides the bandwidth provided by the arbiter among the flows. The arbitration proceeds in two steps. In the first step, a (non-work-conserving) rate decomposition scheme is used to provide isolated bandwidth trunks on every input-output pair. These grants are scheduled among the flows by the GBS scheduler, the delay bound in the first element being dependent on the added latency of the arbiter and the VOS. This has motivated some recent interest [39] in the generation of low-latency smooth templates. In the second step, a maximal matching algorithm is used to maximize throughput, which is distributed by the EBS scheduler in a chosen fair manner. Note that, for admissible traffic, the need for fairness in the distribution of (excess) bandwidth is moot. For traffic that causes (sustained or temporary) instability, excess bandwidth is first divided on an input basis by the arbiter (e.g., using equal weighting in the case of iSLIP), after which it is

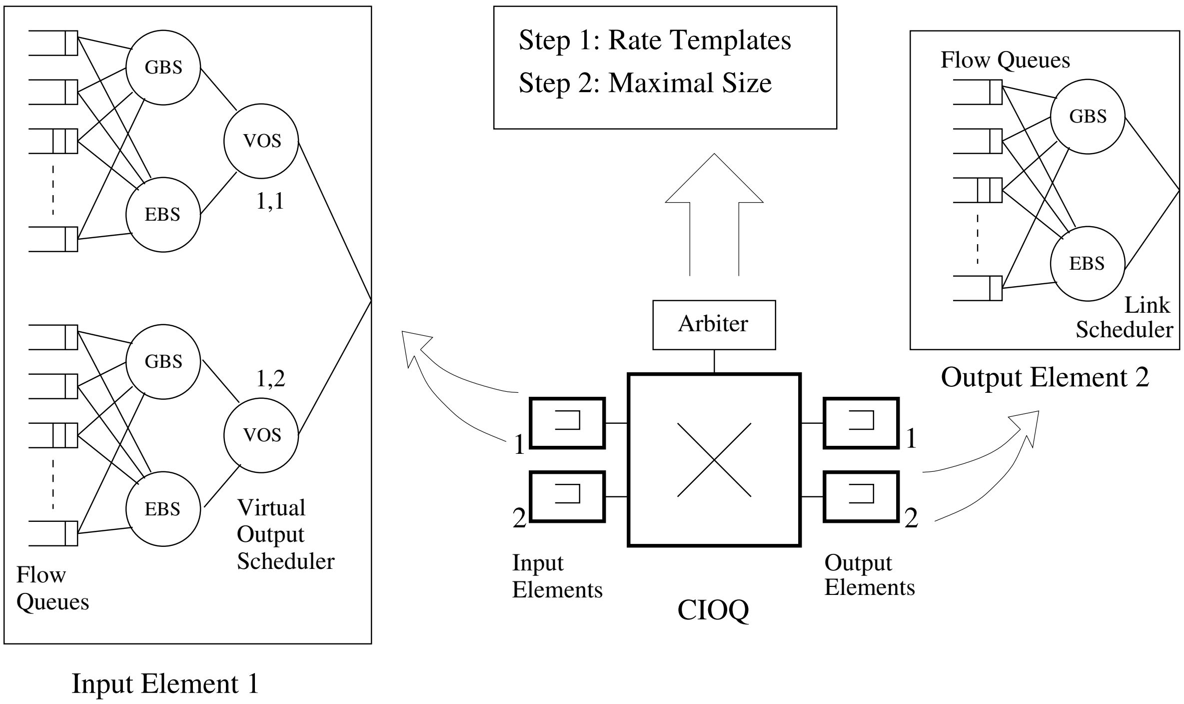


Figure 3.3: QoS in CIOQ using distributed schedulers

distributed by the VOS. Clearly, this does not, in general, correspond to the weighting received by a flow from a flat link scheduler of an OQ switch.

Chapter 4

Buffered Clos switches

The methods for constructing a high capacity switching fabric, composed of smaller space and memory elements, are the subjects of this chapter. We first present the framework of functional equivalence used to characterize the performance of the fabric. The architectures under our consideration are restricted to a class of fully connected three stage models, which we call *Buffered Clos* switches, and are driven by different design constraints. The class includes single-path models such as the CIOQ switch and more general memory-space-memory designs, and multi-path ones such as the parallel packet switch (PPS).

The taxonomy itself is novel, and is unpublished as of this writing. Some of the problems addressed for memory-space-memory switches are still works in progress, for which we present our initial ideas. Regarding multi-path switches, we include our published results as well as a discussion on similar work being conducted elsewhere, and identify some as yet unsolved issues. We conclude by showing how the QoS methodologies, well studied in the context of OQ switches, can be applied to the multi-module Buffered Clos switches.

4.1 Switch Model

A Buffered Clos switch is a three-stage switch with the following structural limitations. The stages are *uniform*, i.e., each stage is composed of a single type of element, memory or space. The switch is *symmetrical* in arrangement, i.e., the number of elements in the third stage, and their inter-connections with the second, are a mirror image of the first. Finally, we restrict ourselves to a *square* switch, i.e., one with an equal number of input and output ports. We introduce the following 5-tuple notation to specify the switch structure:

$$\mathcal{S} : (N, X, P, K, s),$$

where $N \times N$ is the size of the switch, P is the number of first and third stage elements, and K is the number of central stage elements. X is a string that denotes the type of elements in each stage, with $X[i] \in \{S, M\}$, space and memory element, respectively. Due to the property of symmetry, the first stage elements have dimensions of $N/P \times K$, the second $P \times P$, and the third $K \times N/P$. (We assume that N/P is an integer.)

The parameter s refers to the speedup between the first and second stages. Given an external link capacity of R, the total input link bandwidth of the switch is NR. Similarly, if the capacity of the links between the stages is r, the total bandwidth between two adjacent stages is equal to rPK.

- Packet Switches
  - OQ: Output Queued
  - IQ : Input Queued
  - Buffered Clos Switches
    - Single Path
      - CIOQ: Combined Input Output Queued
      - CIOQ-A: CIOQ with Aggregation
      - CIOQ-P: CIOQ with Pipelining
      - G-MSM: General Memory-Space-Memory
    - Multi-Path
      - PPS: Parallel Packet Switch
      - BVN: Birkhoff Von-Neumann

Figure 4.1: A Taxonomy of Buffered Clos switches

By definition, if all the rates are normalized to the external link rate, then the internal capacity can be calculated as $s \cdot \frac{N}{PK}$. Discounting, for now, the effects of bandwidth fragmentation on the internal links, the switch is structurally non-blocking as long as $s \geq 1$. We define a switch as *single-path* if each realizable path between an input and output contains the same memory element(s). If two paths contain different memory elements, the switch is considered *multi-path*. Note that we allow two paths in a single-path switch to consist of different space elements, because a cell experiences a small *fixed* delay as it traverses a space element, essentially exhibiting predictable behavior.

The taxonomy of Buffered Clos switches is obtained by varying X and the relationship between N, P, and K. While we defer the motivation and analysis for the specific points of interest in the taxonomy, we illustrate it in Fig. 4.1, and define the structures as follows: (Strictly speaking, IQ and OQ are not members of this class, but we use the same notation for uniformity.)

OQ	:	$(N, [M], 1, \emptyset, \emptyset)$
IQ	:	$(N, [MS], N, 1, 1)$
CIOQ	:	$(N, [MSM], N, 1, s)$
CIOQ-A	:	$(N, [MSM], P, 1, s)$, N/P is an integer
CIOQ-P	:	$(N, [MSM], N, K, s)$
G-MSM	:	$(N, [MSM], P, K, s)$, N/P is an integer
PPS	:	$(N, [xMx], N, K, s)$, x is variable
BVN	:	$(N, [SMS], 1, K, s)$

We address the performance of all but the last item in the above list, the so-called Birkhoff-Von Neumann (BVN) switch (not to be confused with BVN rate decomposition), which has garnered some recent interest [6, 35] as a general space-memory-space design. The design constraints that lead us through the taxonomy can be ennumerated as follows. The memory element is constrained in size and capacity by the available memory bandwidth, and in complexity by the frequency of link scheduling operations. The space element is constrained in capacity by the transfer rate between an input-output pair, in size by the realizable pin count in ASIC, and in complexity by the frequency of arbitration, which itself depends on the capacity and the size.

As seen from the previous chapter, CIOQ is fairly well-studied and widely deployed. CIOQ-A, CIOQ-P, and G-MSM switches are already found in practice (e.g., [12, 13]), but their performance is relatively less well studied. PPS and BVN are still mainly under theoretical investigation.

4.1.1 Functional Equivalence

We denote a switch $\mathcal{S}_1$ operating under a set of algorithms $\mathcal{A}_1$ to be functionally equivalent with a switch $\mathcal{S}_2$, operating under $\mathcal{A}_2$, as follows:

$$(\mathcal{S}_1, \mathcal{A}_1) \overset{T,f}{\simeq} (\mathcal{S}_2, \mathcal{A}_2),$$

where T is the class of traffic for which the equivalence holds, and f denotes the level of equivalence. The following are the levels which are of our immediate interest:

f_1 Assuming all the average rates $\lambda_{i,j}$ between the input-output pairs are known, and the rates are admissible (i.e., $\forall j\ \sum_i \lambda_{i,j} < 1$), given an algorithm set $\mathcal{A}_2$ which ensures the stability of the queues in $\mathcal{S}_2$, we can find an algorithm set $\mathcal{A}_1$ which ensures the same in $\mathcal{S}_1$. In other words, f_1 denotes the ability to provide virtual trunks of known admissible rates between the input-output pairs.

f_2 Assuming rates $\lambda_{i,j}$ are admissible (i.e., $\forall j\ \sum_i \lambda_{i,j} < 1$), given an algorithm set $\mathcal{A}_2$ which ensures the stability of the queues in $\mathcal{S}_2$, we can find an algorithm set $\mathcal{A}_1$ which ensures the same in $\mathcal{S}_1$, without explicit knowledge of the individual rates. We say that $\mathcal{S}_1$ is relatively stable with respect to $\mathcal{S}_2$.

f_3 Given an algorithm set $\mathcal{A}_2$ which ensures the stability of each output j with admissible rates (i.e., those j for which $\sum_i \lambda_{i,j} < 1$), in switch $\mathcal{S}_2$, we can find a set $\mathcal{A}_1$ which enables the same in $\mathcal{S}_1$. We define an output j to be stable if the system of queues which contain traffic to output j is stable. In other words, f_3 represents the ability of a switch to isolate unstable outputs in the presence of inadmissible traffic. We say that $\mathcal{S}_1$ is relatively stable in the strict sense with respect to $\mathcal{S}_2$.

f_4 Given $\mathcal{A}_2$ which ensures the stability of a subset of input-output pairs with admissible rates (i.e., pairs (i, j) for which $\lambda_{i,j} < w_{i,j}$, the fraction of the output link bandwidth given to the pair by $\mathcal{A}_2$) in switch $\mathcal{S}_2$, we can find $\mathcal{A}_1$ which guarantees the same in $\mathcal{S}_1$ for the same subset. We define an input-output pair (i, j) to be stable if the system of queues which contain the traffic belonging to the pair is stable. f_4 represents the ability of the switch to isolate unstable input-output pairs within the same (unstable) output. We say that $\mathcal{S}_1$ is relatively stable in the strictest sense with respect to $\mathcal{S}_2$.

f_5 Given an algorithm set $\mathcal{A}_2$ for $\mathcal{S}_2$, we can find a set $\mathcal{A}_1$ which ensures that cells depart from $\mathcal{S}_1$ in an exact emulation of the departure from $\mathcal{S}_2$.

In essence, f_1 denotes the ability of a switch to utilize the knowledge of all the admissible rates to ensure that the given rates can be supported. For example, the BVN rate matrix decomposition provides virtual trunks with the given rates for input queued switches, and can be used for f_1 equivalence. Note however that such rate decomposition schemes are stronger than f_1 since they can also guarantee given rate requirements even if the offered arrival rates do not adhere to those requirements. Equivalence at level f_2 has been equated in the literature to providing 100% throughput in packet switches. Levels f_3 and f_4 provide further means to maintain the so-called

100% throughput, while isolating specific flows that contribute to congestion and instability. In that regard, they might also be considered as levels of fairness in the distribution of bandwidth in a (partially) unstable system. Note also that the property of work-conservation, on an output link basis, is sufficient for f_3, but not for f_4 equivalence.

The methodology to prove the stability of a system of queues is left open, which might itself restrict the traffic class T. We establish two important properties to aid in recognizing equivalence. Notice that each level of equivalence subsumes the previous level, i.e., if the equivalence holds at level f_i, then it holds for all $f_j, j < i$. This leads us to the *containment property*:

$$(\mathcal{S}_1, \mathcal{A}_1) \overset{T,f_i}{\simeq} (\mathcal{S}_2, \mathcal{A}_2) \Rightarrow \forall j < i,\ \ (\mathcal{S}_1, \mathcal{A}_1) \overset{T,f_j}{\simeq} (\mathcal{S}_2, \mathcal{A}_2) \tag{4.1}$$

Also, it is easy to see that each level of stability lends itself to the *transitive property*. That is, if a set of equivalences of switch $\mathcal{S}_2$ with respect to $\mathcal{S}_3$ is well known, it suffices to show the equivalence of a new switch $\mathcal{S}_1$ with $\mathcal{S}_2$ in order to establish the same with respect to $\mathcal{S}_3$.

$$\begin{aligned} &(\mathcal{S}_1, \mathcal{A}_1) \overset{T,f}{\simeq} (\mathcal{S}_2, \mathcal{A}_2) \\ \wedge\ &(\mathcal{S}_2, \mathcal{A}_2) \overset{T,f}{\simeq} (\mathcal{S}_3, \mathcal{A}_3) \ \ \Rightarrow (\mathcal{S}_1, \mathcal{A}_1) \overset{T,f}{\simeq} (\mathcal{S}_3, \mathcal{A}_3) \end{aligned} \tag{4.2}$$

Since the optimality of work-conserving OQ switches is well established, we can unambiguously characterize the performance of any given switch $\mathcal{S}$ by finding algorithms $\mathcal{A}$, such that it is functionally equivalent at level f_i with the OQ switch, for the largest value of i and the broadest traffic class T, i.e.,

$$\textbf{Goal}:\ \ (\mathcal{S}, \mathcal{A}) \overset{T,f_i}{\simeq} (\text{OQ, }\{\text{Work Conserving (WC)}\}\ )$$

The known results for IQ and CIOQ switches, which were reviewed in the previous chapter, fits into this framework as follows:

$$\begin{aligned} &(\text{IQ, }\{\text{BVN Decomposition}\}) \overset{T,f_1}{\simeq} (\text{OQ, }\{\text{WC}\}) \\ &(\text{IQ, }\{\text{iSLIP}\}) \overset{T,f_2}{\simeq} (\text{OQ, }\{\text{WC}\}),\ \ T\text{:Bernoulli, i.i.d., uniform; single iteration} \\ &(\text{IQ, }\{\text{iSLIP}\}) \overset{T,f_2}{\simeq} (\text{OQ, }\{\text{WC}\}),\ \ T\text{:i.i.d., uniform; } N \text{ iterations} \\ &(\text{IQ, }\{\text{Maximum Weight}\}) \overset{T,f_2}{\simeq} (\text{OQ, }\{\text{WC}\}) \\ &(\text{CIOQ, }\{\text{Maximal Size}\}) \overset{T,f_2}{\simeq} (\text{OQ, }\{\text{WC}\}),\ \ s = 2 \\ &(\text{CIOQ, }\{\text{CCF}\}) \overset{T,f_5}{\simeq} (\text{OQ, }\{\text{WC, PIFO}\}),\ \ s = 2 \end{aligned}$$

4.2 CIOQ: Some Loose Ends

The CIOQ switch is attractive because the memory bandwidth required is $(1+s)$ times the interface rates, without a dependence on the size of the switch. However, assuming that a large space element with a link rate of s can be built, there are two immediate problems. The first is related to performance. Functional equivalence at level f_5 can be achieved as shown in [17], however the algorithm is exceedingly complex. Practical algorithms have not been shown to be relatively stable in the strict sense. The second problem is related to the frequency of arbitration. The length of an arbitration cycle is given by L/sC, where C is the output link capacity and L is the size of the cell. Therefore, it decreases with increasing transfer rates. Additionally, many of the optical space elements which provide the fastest transfers rate suffer from a high re-configuration latency, once a matching has been computed. We address these issues before proceeding to the next point in the taxonomy.

4.2.1 Strict Relative Stability

One of the desirable properties of an OQ switch is that instability at an output is isolated to that output. In other words, if $\sum_i \lambda_{i,j} \geq 1$ for some output j, it does not affect the stability of queues at other outputs, as long as it employs a work conserving policy. Consider a partitioning of the set of output ports $\{1, 2, \ldots, N\}$ into two proper subsets A and B. Let $\sum_i \lambda_{i,j} < 1$ for all $j \in A$, and $\sum_i \lambda_{i,j} \geq 1$ for all $j \in B$. A CIOQ switch would be equivalent at level f_3 with a work conserving OQ switch, as long as all outputs in A are stable. Of course, directly providing a specific matching algorithm that guarantees work-conservation (such as the one in [44]) immediately establishes f_3 equivalence. However, our goal here is to provide the same using simpler matchings.

Note that each input element i has virtual output queues corresponding to outputs in both A and B. Let $Q_{i,j}$ and $Q_{i,k}$ be two such queues, respectively. At first glance, it seems that the combination of the instability of $Q_{i,k}$ and a bad maximal size matching algorithm which always chooses that queue would cause the instability to spread to $Q_{i,j}$. However, we conjecture[1] that because of the physical input link limitation $\sum_j \lambda_{i,j} \leq 1$, any maximal matching with speedup of 2 is sufficient to ensure relative stability in the strict sense. (This strengthens the result proved in [19], which applied only when all the rates were admissible.)

Conjecture 1 *(CIOQ, $\{$Maximal Size$\}$)* $\overset{T,f_3}{\simeq}$ *(OQ, $\{$WC$\}$),* $s = 2$

Next, consider an output j belonging to the set B of unstable ports. Also, assume that the reference OQ switch has the ability to divide the capacity of output j according to normalized weights $w_{i,j}$ (e.g., by using *virtual input* queueing at the outputs, and any weight-based scheduling scheme) among the pair (i, j). Then, if $\lambda_{i,j}$ turns out to be less than $w_{i,j}$, the pair (i, j) remains stable, even within an unstable port. In other words, a work conserving OQ switch with any weight based policy has the ability to isolate instability on an input-output pair basis. Our next goal is to enable the same in a CIOQ switch, thereby providing f_4 equivalence.

Consider a CIOQ switch with virtual input queues $Q'_{i,j}$ in each output element j for each of the inputs. We can show by a simple counter example that *any* maximal matching algorithm cannot ensure relative stability with an OQ switch in the strictest sense. Let $\lambda_{1,j} = \lambda_{2,j} = 1$, and $\lambda_{3,j} = 1/2$, with no other traffic in the switch, and weight $w_{3,j} = 1/2 + \delta$. Clearly, in the reference OQ switch, the pair $(3, j)$ would remain stable. However, irrespective of the scheduling scheme among the virtual input queues of the CIOQ, a bad maximal matching can always choose pairs $(1, j)$ and $(2, j)$ in an alternating fashion, and leave out $(3, j)$ altogether even with a speedup of 2.

To remedy the above, we propose the *Shortest Output Queue First* (SOQF) algorithm, which works as follows. In the output elements, the virtual input queues are scheduled with the same weights $w_{i,j}$ as in the reference OQ switch. The maximal matching proceeds by first sorting the virtual input queue lengths, and in ascending order, matching the first pair (i, j) with unmatched i and j, which has a cell to send. It can be easily shown that this is an instance of a maximal size

[1] This result is somewhat counter-intuitive, and a tremendous effort to find a counter-example turned out to be futile. Of course, we need a rigorous proof to establish this as a theorem. An author of the original result on maximal size seems to concur with this conjecture [59].

matching, and hence is at least equivalent to a work-conserving OQ switch at level f_2 (and f_3 if Conjecture 1 is proved). We claim[2] that by giving preference in the maximal matching to the least congested pair, a CIOQ switch can isolate instability on an input-output pair basis. The running time of SOQF is dominated by the time to sort the output queues, i.e., $O(N^2 \log N)$.

Conjecture 2 *(CIOQ, $\{$SOQF$\}$)* $\overset{T,f_4}{\simeq}$ *(OQ, $\{$WC, Weight Scheduling$\}$),* $s = 2$

4.2.2 Envelope and Batch Switching

There have been two recent approaches to tackle the problems associated with a high arbitration frequency. The first [36] recognizes the fact that the frequency is given by sC/L and can be reduced by increasing the denominator. Several cells are packed into an envelope and a matching is computed on an envelope timeslot. It has been proven that this does not affect the stability of the queues (each queue will have an additional length of at most an envelope length), however, since envelopes are released only when full, the delay of a cell in a partial envelope is potentially unbounded. An additional speedup (of upto 2) s_1 can be used to bound the delay. Note that envelopes also enable moderately complex (and distributed) matching algorithms by amortizing the complexity over the envelope timeslot.

Another, possibly complimentary, approach [70] deals with a common problem associated with optical elements, that of high re-configuration times. Even though matchings may proceed at a frequency of sC/L, where L is now the envelope size, these may be used in conjunction with even slower re-configuration, by essentially accumulating a batch of matchings and computing a sequence of configurations that cover the batch. In [70], it has been shown that $2N$ configurations with an additional speedup of $s_2 = 2$ is sufficient to cover a batch of any size. Though we do not propose any new algorithms that address these issues, we need to note that the actual speedup of a space element, in practice, would equal $s_1 s_2 s$, where s is the speedup of a CIOQ switch that does not employ either of the two techniques.

4.3 Single-Path Buffered Clos Switches

The CIOQ switch is the simplest instance of a single-path Buffered Clos switch. We now show how to add the techniques of aggregation and pipelining to obtain more complex designs, and the algorithms that maintain equivalence with an OQ switch.

4.3.1 CIOQ-A: Aggregation

A CIOQ switch with aggregation (CIOQ-A) is obtained by grouping N/P consecutive interfaces of the former into the same memory element, as shown in Fig. 4.2(b). Consequently, there are P memory elements in the first and third stage, and a single space element in the second stage. Each memory element now requires a memory bandwidth of $(1 + s)N/P$ times the interface rate. The space element requires a transfer rate of sN/P on each link (instead of s in CIOQ) and an

[2] As of now, we possess the intuition behind the algorithm, but no analytical proof. If we fail in the latter, the plan is to validate the algorithm by resorting to simulations. We may also use the scheme to provide equivalence at level f_3 in case the previous conjecture turns out to be incorrect or in the case when the sum of the incoming rates into an input element exceeds unity due to multicast traffic.

arbitration and configuration frequency of sN/P matchings per timeslot. We refer to the cycle time of transfer of the space element as the *internal* timeslot, which is P/sN times the external timeslot. From a design point of view, this approach has a few advantages. The size of the space element, and hence the number of inter-connections and the internal contention[3] points, decreases by a factor of P. A maximal match algorithm on the smaller space element has a time complexity of $s\frac{N}{P}O(P^2)$ per external timeslot, or $O(NP)$ as opposed to $O(N^2)$ in CIOQ (though this point is rendered ineffective in some of the more complex algorithms explained below). Aggregation also allows the reuse of existing space elements to support multiple subports. The primary disadvantage of this approach is the higher memory bandwidth compared to a CIOQ switch of the same dimensions and the higher frequencies of scheduling in the memory elements and arbitration for the space element.

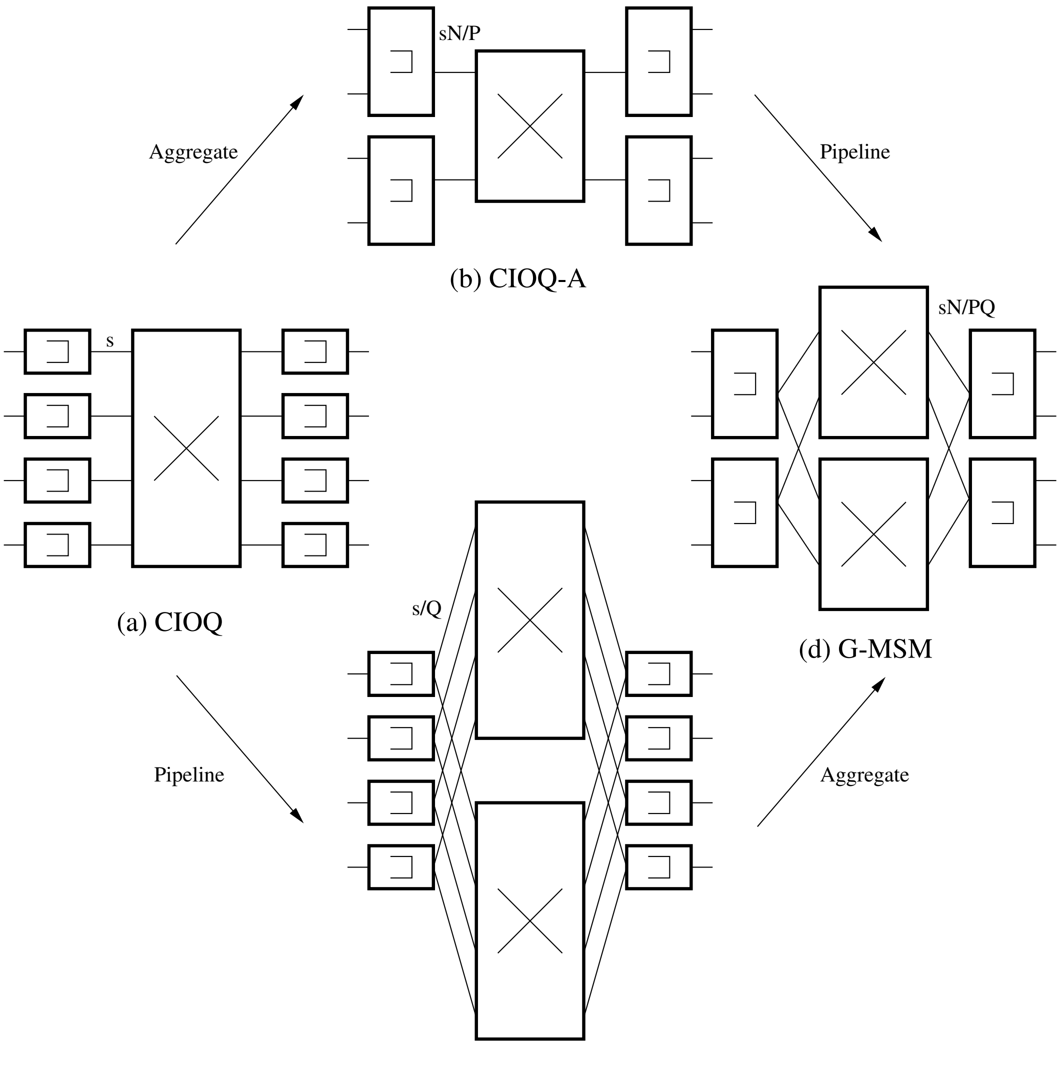


Figure 4.2: Single Path Buffered Clos Switches

The same rate decomposition techniques of CIOQ, as reviewed in Sec. 3.2.2 can be performed

[3]The advantages of lower number of contention points will likely not be revealed in maximal matchings with worst-case speedup, but might be apparent in sub-maximal matchings and lower speedup. This needs to be verified.

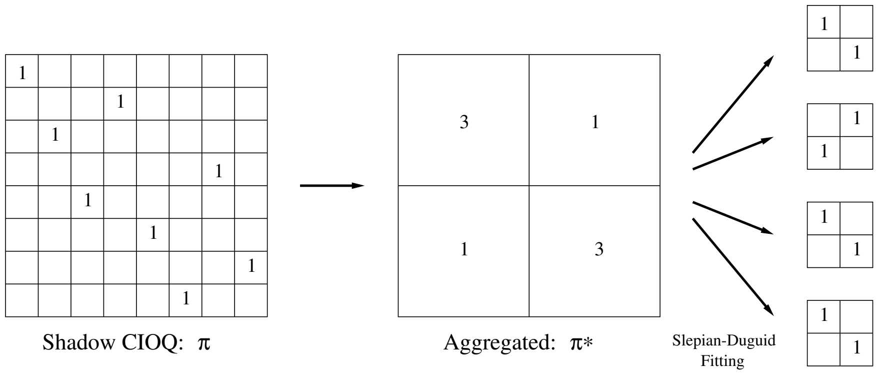


Figure 4.3: Example: To shadow a CIOQ switch

on an input-output element basis, for known admissible rates to achieve equivalence at level f_1 with an OQ switch. Furthermore, a maximal size matching done an *element pair* basis, can be easily shown to be relative stable with OQ for $s = 2$. The straightforward proof of the following may be found in Appendix C.1. (Note that f_3 equivalence also holds if Conjecture 1 is true.)

Theorem 1 *(CIOQ-A, $\{$Maximal Size$\}$)* $\overset{T,f_2}{\simeq}$ *(OQ, $\{$WC$\}$)*, $s = 2$

For stronger equivalence, we propose an algorithm called *shadow CIOQ-$\mathcal{A}$*, which emulates a CIOQ switch running $\mathcal{A}$, followed by a decomposition of the matchings, to achieve the same level of equivalence with an OQ switch as that achieved by $\mathcal{A}$. Consider an $N \times N$ CIOQ-A, with all rates normalized with respect to the interface rates. Let the queues be arranged (e.g., a combination of virtual output and virtual input queues) as required by $\mathcal{A}$. Shadow CIOQ-$\mathcal{A}$ first computes s^* matchings every timeslot by running $\mathcal{A}$, where s^* is the speedup required to operate $\mathcal{A}$ in a CIOQ switch with the same dimensions. Consider one such matching π, and compose an aggregate matching π^* as follows:

$$\forall i \leq P,\ \ j \leq P \quad \pi^*_{i,j} = \sum_{k=\frac{N}{P}(i-1)+1}^{k \leq \frac{N}{P}i} \ \sum_{l=\frac{N}{P}(j-1)+1}^{l \leq \frac{N}{P}j} \pi_{k,l}$$

Note that the aggregate matching has the following properties. The sum of all entries in a column or a row is no greater than N, and each entry sums up to no greater than N/P. Viewing the entries as number of circuits, it follows from Clos theorem (Sec. 3.1) that π^* can be partitioned into $2(N/P) - 1$ matchings in a greedy fashion in $O(\frac{N^2}{P})$ time, or using the Slepian-Duguid theorem into N/P matchings in $O(NP^2)$ time. Each of the computed partitions is used as a matching for the $P \times P$ space element, running at at a frequency of sN/P. Consequently, the shadow CIOQ-$\mathcal{A}$, followed by Clos or Slepian-Duguid (SD) fitting, requires a speedup of $(2 - \frac{P}{N})s^*$ or s^*, respectively, to exactly emulate a CIOQ switch running $\mathcal{A}$ at a speedup of s^* (see Fig. 4.3 for an example). We have just proved the following result:

Theorem 2

(CIOQ-A, $\{$Shadow CIOQ-$\mathcal{A}$, Clos Fitting$\}$) $\overset{T,f_5}{\simeq}$ *(CIOQ, $\mathcal{A}$)*, $s = (2 - \frac{P}{N})s^*$

(CIOQ-A, $\{$Shadow CIOQ-$\mathcal{A}$, SD Fitting$\}$) $\overset{T,f_5}{\simeq}$ *(CIOQ, $\mathcal{A}$)*, $s = s^*$

The above result, along with the containment and transitive properties of functional equivalences, leads us to the following:

Corollary 1 *(CIOQ, $\mathcal{A}$)* $\overset{T,f_i}{\simeq}$ *(OQ, $\mathcal{A}'$),* $s = s^* \quad \Rightarrow$
(CIOQ-A, $\{$Shadow CIOQ-$\mathcal{A}$, Clos Fitting$\}$) $\overset{T,f_i}{\simeq}$ *(OQ, $\mathcal{A}'$),* $s = (2 - \frac{P}{N})s^*$, *and*
(CIOQ-A, $\{$Shadow CIOQ-$\mathcal{A}$, SD Fitting$\}$) $\overset{T,f_i}{\simeq}$ *(OQ, $\mathcal{A}'$),* $s = s^*$

This allows us to establish equivalences for CIOQ-A by applying well known results for CIOQ. For example, a CIOQ-A switch is relatively stable in the strictest sense with an OQ switch, by shadowing a CIOQ with the SOQF algorithm with speedup 2, followed by fitting. Clos fitting requires a total speedup of $4 - 2\frac{P}{N}$ and a complexity of $O(N^2 \log N)$ dominated by the sorting time, while SD fitting requires a total speedup of 2 and a complexity of $O(N^2 \log N + NP^2)$. An open question is whether equivalence at level f_4 (i.e., strictest sense) can be established without resorting to a shadow CIOQ, possibly in faster time.

4.3.2 CIOQ-P: Pipelining

A CIOQ switch with pipelining (CIOQ-P) is obtained by replacing a single space element of CIOQ, running at interface rate s, by K instances of elements running at rate s/K, as shown in Fig. 4.2(c). Each memory element requires a memory bandwidth equal to $(1 + s)$ times the external interface rates (same as in CIOQ). Each space element has a configuration frequency of s/K matchings per timeslot. The length of an internal timeslot now is much longer, equal to K/s times the external timeslot. The frequency of the total number of arbitrations, over all the space elements, remains at s per timeslot. The main design advantage of this approach is that slower space elements of the same dimensions may be used to construct a higher interface rate switch. Also, if the matchings for all the elements can be computed in parallel (a non-trivial task), the frequency of arbitrations goes down by a factor of K.

We first show two algorithms that rely on shadowing a CIOQ switch, in the same fashion as in the previous section. Again, let s^* be the speedup required by algorithm $\mathcal{A}$ on CIOQ, to achieve a certain equivalence with an OQ switch. Shadow CIOQ-$\mathcal{A}$ computes matchings $\pi(n)$ at a frequency of s^* per timeslot. Each space element k is configured with a matching $\pi^{(k)}(n)$ where

$$\pi^{(k)}(n) = \pi(K(n-1) + k)$$

Therefore, each space element gets $1/K$ of the original matchings, which can be satisfied as long as $s = s^*$. Since $\pi(n)$ is essentially allocated to the K space elements, one element at a time, we call this method *sequential dispatch*.

To bring down the arbitration frequency in line with the lower transfer rates of the space element, we propose a method called *striping*. In this, we shadow a CIOQ switch running $\mathcal{A}$ with envelopes of size K. If $s^{*'} \geq s^*$ is the speedup required to achieve the desired equivalence while allowing for the effects of the envelopes [36], then the frequency of arbitration is brought down to $s^{*'}/K$ envelope matchings per timeslot. For each such matching, we break the envelope and assign a single cell to each of the K space elements. The assignments can be satisfied as long as $s = s^{*'}$. Essentially, we have proved the following results:

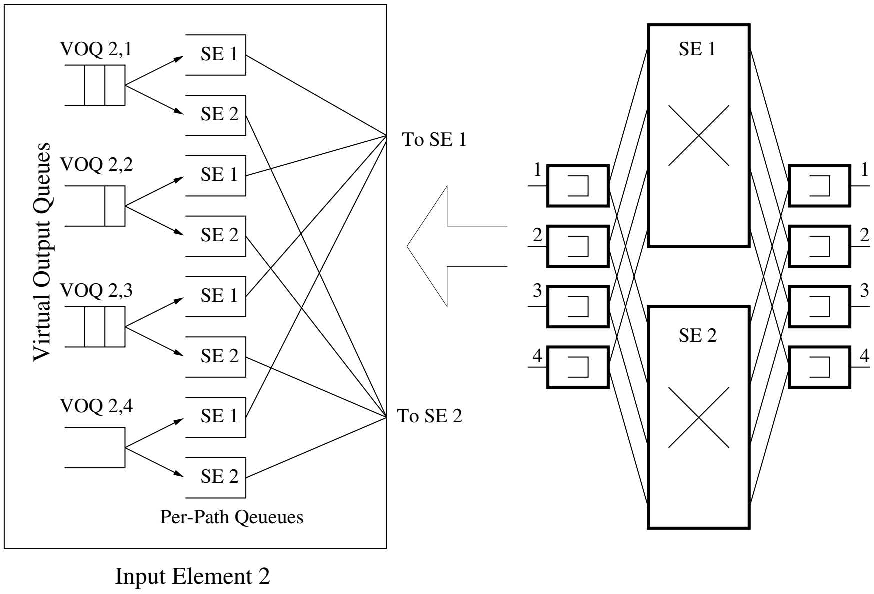


Figure 4.4: Equal Dispatch using per-path queues

Theorem 3

$$\textit{(CIOQ-P, }\{\textit{Shadow CIOQ-}\mathcal{A}\textit{, Sequential Dispatch}\}\textit{)} \stackrel{T,f_5}{\simeq} \textit{(CIOQ, }\mathcal{A}\textit{)}, \quad s = s^*$$
$$\textit{(CIOQ-P, }\{\textit{Shadow CIOQ-}\mathcal{A}\textit{, Striping}\}\textit{)} \stackrel{T,f_5}{\simeq} \textit{(CIOQ, }\mathcal{A} \bigcup \{\textit{K-Envelopes}\}\textit{)}, \quad s = s^{*'}$$

Again, by using the containment and transitive properties of equivalences, we have the following result for proving a desired equivalence of CIOQ-P with an OQ switch: (and a similar one for the striping algorithm)

Corollary 2 $\quad \textit{(CIOQ, }\mathcal{A}\textit{)} \stackrel{T,f_i}{\simeq} \textit{(OQ, }\mathcal{A}'\textit{)}, \quad s = s^* \quad \Rightarrow$

$$\textit{(CIOQ-A, }\{\textit{Shadow CIOQ-}\mathcal{A}\textit{, Sequential Dispatch}\}\textit{)} \stackrel{T,f_i}{\simeq} \textit{(OQ, }\mathcal{A}'\textit{)}, \quad s = s^*$$

It would be advantageous to have methods which enable equivalences without resorting to either shadowing CIOQ, or using envelopes of size K. Our goal is to maintain an arbitration frequency of s/K using concurrent arbiters attached to each space element. Recognize that envelopes, in essence, allow to distribute the matched traffic in an exactly equal proportion to each of the K elements. We propose an algorithm called *equal dispatch* which achieves the same result using a slightly complex queueing logic. In each input memory element, per-path virtual output queues are maintained. That is, in element i, there is a VOQ $Q_{i,j}^{(k)}$ for output j, for each of the space elements k, as shown in 4.4. The traffic belonging to an input-output pair (i, j) is split equally (on a round robin basis) to each $Q_{i,j}^{(k)}$, $k = 1, \ldots, K$. Each space element k computes matchings concurrently based only on the state of the per-path queues corresponding to k. Essentially, per-path queueing allows to distribute the traffic belonging to (i, j) to K concurrent planes, each being served by an individual space element.We can show (see Appendix C.1) that if each element performs maximal matching, all the queues are stable for admissible traffic, as long as $s = 2$.

Theorem 4 $\quad \textit{(CIOQ-P, }\{\textit{Equal Dispatch, Maximal Size}\}\textit{)} \stackrel{T,f_2}{\simeq} \textit{(OQ, }\{\textit{WC}\}\textit{)}, \quad s = 2$

Note that the above would also hold for f_3 equivalence if Conjecture 1 is true. Similarly, if each output memory element maintains per-path virtual input queues, grouped on the basis of the space element that feeds into it, we can easily show that f_4 equivalence also holds, as long as Conjecture 2 is true. The output scheduler operates using weight $w_{i,j}/K$ for virtual input queue $Q_{i,j}^{(k)'}$, where $w_{i,j}$ is the weight given to the input-output pair (i, j) by a scheduler in the reference OQ switch.

Conjecture 3 *(CIOQ-P, {Equal Dispatch, SOQF})* $\overset{T,f_4}{\simeq}$ *(OQ, {WC, Weight Sched}),* $s = 2$

For implementation purposes, to avoid mis-sequencing, the per-path queues are not physically separate. Indeed, all the cells are enqueued into the same physical virtual output (and input) queue, and only the per-path *occupancies* are maintained. These occupancies are viewed as actual queue lengths for matching purposes only. By sequentially numbering the space elements, or by slightly skewing the cell transfers, the ordering of the cells can be determined at the output element.

4.3.3 G-MSM: Memory Space Memory

A General Memory-Space-Memory (G-MSM) switch may be constructed by combining aggregation and pipelining as shown in Fig. 4.2(d). Each memory element requires a memory bandwidth of $(1 + s)N/P$ times the interface rates. Each space element has a transfer rate and configuration frequency of sN/PK. The total arbitration frequency, over all the space elements, is sN/P matchings per timeslot, as in CIOQ-A. The length of an internal timeslot of the space element is PK/sN times that of the external timeslot, and may be higher or lower depending on the chosen dimensions.

The performance of a G-MSM switch is a natural combination of that of the CIOQ-A and CIOQ-P designs. Consider a G-MSM that employs VOQs $Q_{i,j}^{(k)}$ in input element i for each path (space element) k, where j is the destination output element. We can show, similar to Theorems 1 and 4 that the equal dispatch algorithm followed by K concurrent maximal size matchings, at a frequency of $2N/PK$, is sufficient to ensure relatively stability with a work-conserving OQ switch:

Theorem 5 *(G-MSM, {Equal Dispatch, Maximal Size})* $\overset{T,f_2}{\simeq}$ *(OQ, {WC}),* $s = 2$

While the truth of Conjecture 1 will allow to easily establish f_3 equivalence, an open question is whether equivalence at level f_4 can be established using a variation of SOQF, without resorting to shadowing.

There are three shadowing approaches for G-MSM, which may be used to achieve stronger equivalences. In the first, virtual input (and output) queues are maintained, as required by a target algorithm $\mathcal{A}$, on a per-path basis. The equal dispatch algorithm is used to separate traffic into K planes. Each path then (in parallel) shadows a CIOQ switch at a frequency of s^*/K matchings per timeslot, followed by the aggregation and fitting techniques of CIOQ-A. While equal dispatch does not lend itself to an exact emulation of a CIOQ switch, the shadowing and fitting algorithms emulate a CIOQ in each plane (Theorem 2). Combining with Conjecture 3, we obtain:

Conjecture 4
(G-MSM, {Equal Dispatch, Shadow CIOQ-$\mathcal{A}$, Clos Fitting}) $\overset{T,f_4}{\simeq}$ *(CIOQ, $\mathcal{A}$),* $s = (2 - \frac{P}{N})s^*$,
(G-MSM, {Equal Dispatch, Shadow CIOQ-$\mathcal{A}$, SD Fitting}) $\overset{T,f_i}{\simeq}$ *(CIOQ, $\mathcal{A}$),* $s = s^*$

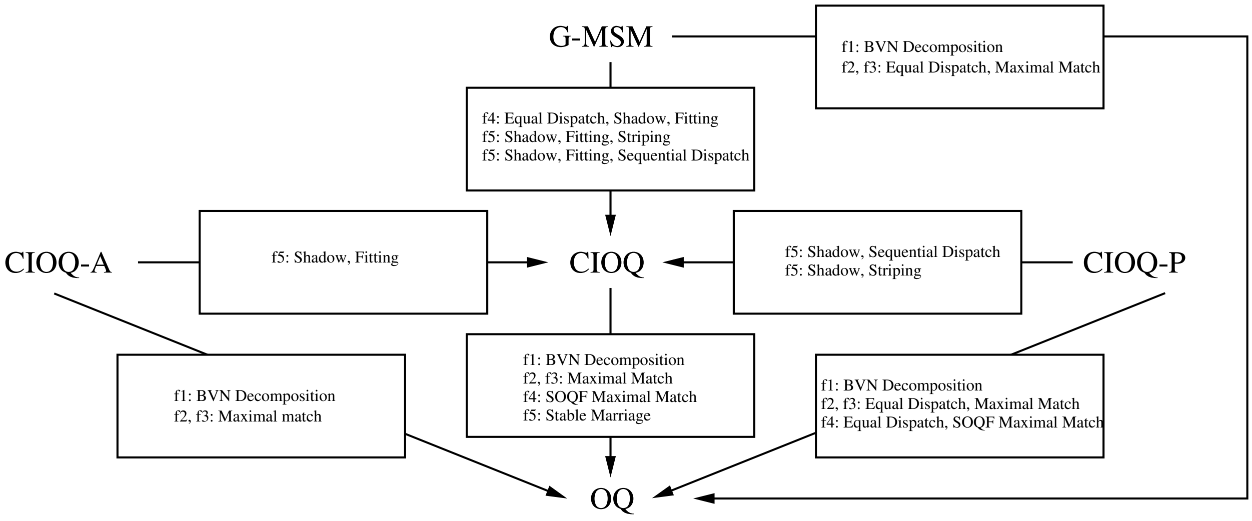


Figure 4.5: Equivalences for single-path Buffered Clos switches

The above may be extended by the transitive property to achieve any equivalence $f_i, i \leq 4$ with a reference OQ switch.

The second approach is an extension of the striping algorithm, which does away with the need for equal dispatch or per-path queues. Choosing an envelope size of K cells and maintaining a shadowing frequency of $s^{*'}/K$, the matchings are aggregated and partitioned in a manner similar to the one in CIOQ-A. The same partitions are used to configure all the K space elements in each internal timeslot. Each of the space elements then carries a single cell from the envelope. Since striping (Theorem 3) and aggregating (Theorem 2) provide for precise emulation of a CIOQ, we obatin the following:

Theorem 6
(G-MSM, $\{$Shadow, Clos Fitting, Striping$\}$) $\overset{T,f_5}{\simeq}$ *(CIOQ, $\mathcal{A} \bigcup \{$K-Envelopes$\}$),* $\quad s = (2 - \frac{P}{N})s^{*'}$,
(G-MSM, $\{$Shadow, SD Fitting, Striping$\}$) $\overset{T,f_5}{\simeq}$ *(CIOQ, $\mathcal{A} \bigcup \{$K-Envelopes$\}$),* $\quad s = s^{*'}$

The third approach is an extension of the sequential dispatch algorithm, and requires a shadowing frequency of $s^*(\leq s^{*'})$ matchings per timeslot, but does not use envelopes. Each of the matching is aggregated, as in a CIOQ-A, however, the resulting partitions are used to configure one space element after the other in a sequential fashion. Note that the number of partitions (N/P if SD fitting is used) is not required to exactly fit in the K elements. We obtain the same result as in the above theorem, with $s^{*'}$ replaced by s^*. The results corresponding to the last two approaches may be extended, again using the transitive property (similar to Corollary 2), to achieve the desired equivalence with an OQ switch.

Fig. 4.5 summarizes all the equivalences discussed so far (including the conjectures). The containment and transitive properties allow to follow multiple hops of the equivalence graph. An interesting open problem for a G-MSM switch is whether the memory elements can be recursively (!!) constructed using a $N/P \times K$ CIOQ switch. This seems to be possible for all equal dispatch-based algorithms, however, with physically separate per-path queues, which puts it in the realm of multi-path switches.

4.4 Parallel Packet Switches

A parallel packet switch (PPS) allows to pool the bandwidth resources on several switching paths in order to construct a high capacity switch. As shown in Fig. 4.6, an $N \times N$ switch with interface rate R is composed of K logical memory elements, each of which is of size $N \times N$ and operates at an interface rate of r. A central element is referred to as a *core* switch, while the $1 \times K$ first stage element is referred to as an ingress demultiplexor and the $K \times 1$ third stage element is referred to as an egress multiplexor. The K core switch elements may be realized by a single physical $KN \times KN$ memory element, in which case the switch is referred to as an *inverse multiplexed* switch [10], or by K separate physical planes [33, 42]. Also, the memory element(s) itself may be replaced by a CIOQ switch, which is equivalent at a given level f_i to an OQ switch (a single memory element). For the purposes of analyses however, we will restrict ourselves to the multi-plane realization, and K physical memory elements, with the understanding that an f_i equivalence with an OQ switch for this system will continue to hold if the memory elements are replaced with a CIOQ exhibiting the same level of equivalence[4].

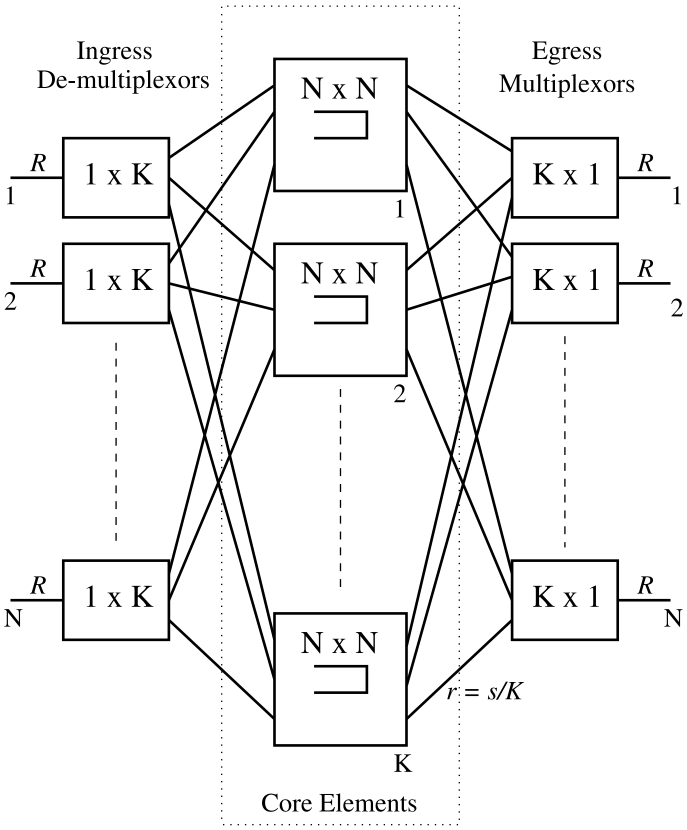


Figure 4.6: A parallel packet switch (PPS)

If s is the speedup of the switch, then the internal links operate at s/K times the interface rates. Assuming a CIOQ implementation with speedup s^* for the core switch, the memory bandwidth required in the core is $(1 + s^*)s/K$ times the interface rate, which can be made significantly lower

[4]This statement itself needs a rigorous proof, which we omit here.

than the interface rate by choosing an appropriate K. However, if the first and third stages are also composed of memory elements, they still require a memory bandwidth of $(1+s)$ times the interface rates. Thus, we identify two significant benefits of this architecture. Firstly, it allows to reuse lower capacity components, even though it might require the design of memory elements running at rates comparable to the higher target interface rates. Secondly, it raises the possibility of using space elements in the first and third stages, thereby allowing to build a switch in which the fastest memory runs at a fraction of the external interface rates. Our goal is to establish equivalences for the PPS, while attempting to keep the ingress and egress either memoryless or with a small fixed amount of memory which can be implemented on-chip.

4.4.1 Flow-Based PPS

Consider a PPS with $X = [MMM]$, i.e., all the stages are composed of memory elements. Assuming the rates of all the flows that traverse the switch are known, f_1 equivalence with an OQ switch may be achieved by statically assigning a flow to a switching path in one of the K core elements. The resulting switch is called a *flow-based* PPS. A path for an admissible flow can be found as long as the number of paths satisfy the condition for multirate Clos topologies (see Sec. 3.1), which translates to the following result:

Theorem 7 *(PPS, {Clos Fitting})* $\overset{T,f_1}{\simeq}$ *(OQ, {WC}),* $X = [MMM], K \geq 2 \left\lfloor \frac{R-b_{\max}}{r-b_{\max}+b_{\min}} \right\rfloor + 1$

Note that per-path queues are required in the ingress and egress elements to buffer the burstiness of the flows assigned to the respective paths. The problem associated with the fragmentation of the internal link bandwidth can be mitigated by *splitting* a selected number of high bandwidth flows across multiple paths, and assigning weights to each path for the split flows [10]. The advantage of static assignment is that there is no mis-sequencing of packets (except for the split flows), as there is a single fixed path for each of the flows. Several so-called *clustered* routers use this approach due to its simplicity. Even flows with no specified allocations are assigned a path, usually through uniform hash functions. Flow-based PPS has several disadvantages. Bufferless elements in the first and third stages cannot be contemplated. In addition, a large number of high bandwidth (and hence, split) flows essentially negates the sequencing advantage. Lastly, it can be shown by a simple counter-example[5] that equivalence with an OQ switch at level f_2 or higher cannot be achieved.

4.4.2 Packet-by-Packet PPS

A *packet-by-packet* PPS performs the path assignment on a per-packet basis in contrast to static flow assignment. Let $\lambda_{i,j}$ be the average rate of the traffic between the input-output pair (i, j). Consider an *equal dispatch* algorithm that distributes traffic belonging to the pair equally among all the K paths offered by the core elements, either on a round robin basis for fixed sized units (cells), or using deficit round robin for variable sized packets. In other words, each flow, defined as an input-output pair in this case, is distributed equally among all the paths. Consequently, the

[5]Consider the case when the flows assigned to a certain path k, and destined to output j, generate more than s/K times the total average rate destined to that output. Even when $\sum_i \lambda_{i,j} < 1$, it causes instability in path k, while some other path(s) is underutilized.

average rate of the traffic between the pair (i, j) in the kth core element is given by

$$\forall k \quad \lambda_{i,j}^{(k)} = \frac{1}{K}\lambda_{i,j}$$

Therefore, if output j is stable in a reference OQ switch, i.e., $\sum_i \lambda_{i,j} < R$, we have $\sum_i \lambda_{i,j}^{(k)} < R/K$ for each of the core elements. A work conserving core element with interface rate R/K, i.e., $s = 1$, ensures that output j is also stable in that element. Thus, we have the following result:

Theorem 8 *(PPS, $\{$Equal Dispatch$\}$)* $\overset{T,f_3}{\simeq}$ *(OQ, $\{$WC$\}$),* $X = [MMM], s = 1$

The above result continues to hold when the granularity of equal dispatch is smaller than an input-output pair, e.g., an equal dispatch of individual user flows among the paths. Per-path queues are required for each split flow in the ingress element to resolve contention among many flows that simultaneously choose the same path, while similar queues are required in the egress element to correctly order the packets which arrive from different paths. Since traffic is equally split atleast on an input-output pair basis, Theorem 8 can be easily extended by considering weight-based schedulers in the core elements with exactly the same weights as in the reference OQ switch.

Corollary 3 *(PPS, $\{$Equal Dispatch, Weight Sched$\}$)* $\overset{T,f_4}{\simeq}$ *(OQ, $\{$WC, Weight Sched$\}$),* $X = [MMM], s = 1$

We will now extend the equal dispatch algorithm to a cell-based *fractional dispatch* algorithm [42], which enables a PPS to have bufferless ingress elements, while maintaining the same equivalence. First, we make two observations regarding the above results. A fluid model PPS which divides the arriving traffic equally, in an instantaneous fashion to the K paths, satisfies Theorem 8. Moreover, no buffers are required in the ingress due to the instantaneous dispatch, and none are required in the egress since all the core elements behave as mirror images of each other. Secondly, notice that given an interface rate R and an internal rate r, the number of required core elements is given by $K = \lceil R/r \rceil$. While an equal dispatch algorithm is sufficient for relative stability in a cell-based PPS model, a fractional dispatch which ensures that out of $\lceil R/r \rceil$ consecutive cells belonging to a pair (i, j), not more than one cell is forwarded to any single core element, is also sufficient for the same (See Appendix C.2).

Theorem 9 *(PPS, $\{$Fractional Dispatch$\}$)* $\overset{T,f_3}{\simeq}$ *(OQ, $\{$WC$\}$),* $X = [MMM], s \geq \frac{\lceil R/r \rceil}{R/r}$

Next, we recognize that when a cell arrives at an input interface and is assigned to a path, the path remains unusable for $\lceil R/r \rceil - 1$ additional timeslots, as shown in Fig. 4.7. Therefore, an arriving cell finds a total of at most $\lceil R/r \rceil - 1$ busy interfaces to the core elements. In addition, if the ingress performs fractional dispatch, the $\lceil R/r \rceil - 1$ previously used paths cannot be used to dispatch the current cell. However, the cell can be immediately dispatched, without queueing, if both the conditions are simultaneously satisfied in the worst-case, i.e, if and only if,

$$\begin{aligned} K &\geq (\lceil R/r \rceil - 1) + (\lceil R/r \rceil - 1) + 1 \\ &= 2\left\lceil \frac{R}{r} \right\rceil - 1 \end{aligned}$$

This translates to a speedup of $s = K/\lceil K/2 \rceil$. Furthermore, we showed in [42] that fractional dispatch has the same properties even when it is carried out over large block of cells, thereby

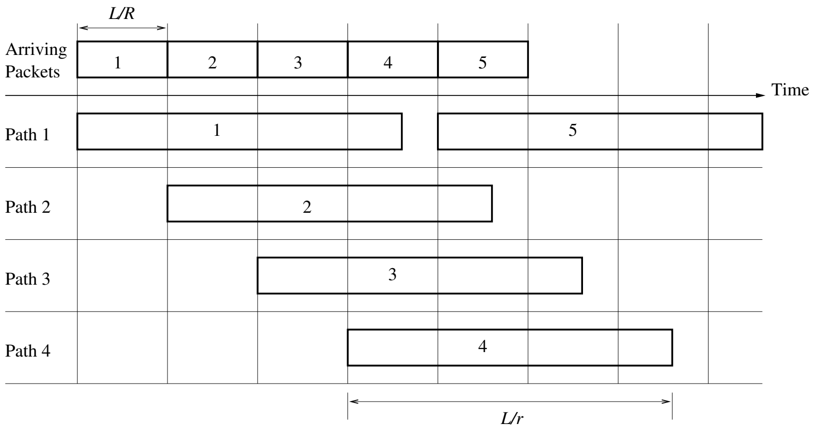


Figure 4.7: PPS: Timeslot Structure, $\lceil R/r \rceil = 4$

allowing its application to variable packet sizes. Specifically, the algorithm stipulates that out of a block of q cells, no more than p are dispatched to the same path, where

$$\frac{1}{\lceil \frac{R}{r} \rceil} \leq \frac{p}{q} \leq \frac{r}{R} \text{ and } \frac{p+1}{q} > \frac{r}{R}$$

Since the condition of Theorem 9 is not violated by the instantaneous fractional dispatch, we have the following result for a PPS with bufferless ingress elements:

Theorem 10 *(PPS, $\{$Fractional Dispatch$\}$)* $\overset{T,f_3}{\simeq}$ *(OQ, $\{$WC$\}$),* $X = [SMM], s = \frac{K}{\lceil K/2 \rceil}$

Note that even though the element is bufferless, there do exist some on-chip buffers to perform header processing, and to execute the load balancing logic. Also, with fractional dispatch, the egress element still requires buffers for re-sequencing. A work item in progress is to find the mis-sequencing bound for this algorithm, and hence the size of the egress memory in order to ascertain if it is small enough to be implemented on-chip (therefore qualifying the egress also as a space element). Another item [41] is the trade-off between buffers and speedup, while using fractional dispatch to maintain relative stability.

S. Iyer *et al.* [33] have shown that an OQ switch with a FIFO discipline can be exactly emulated by a PPS with bufferless ingress and egress elements, with the same speedup as in the last theorem. The proposed emulation algorithm, called the *centralized PPS algorithm* (CPA), maintains a shadow OQ switch, and for every incoming cell at time n for the pair (i, j), determines the time $D(n, i, j)$ when the cell departs in the shadow switch. Recognizing that at most $\lceil R/r \rceil - 1$ paths at the ingress i will be busy at the arrival time n, and a similar number at egress j at the departure time $D(n, i, j)$, a path which simultaneously satisfies both the input and output constraints can be found as long as $K \geq 2\lceil R/r \rceil - 1$. Therefore, we have

Theorem 11 *(PPS, $\{$CPA$\}$)* $\overset{T,f_5}{\simeq}$ *(OQ, $\{$WC, FIFO$\}$),* $X = [SMS], s = \frac{K}{\lceil K/2 \rceil}$

A followup result [34] showed that a distributed variant of CPA, which maintains the input and output constraints independently at each input, can emulate an OQ switch within a delay of $\lceil N/s \rceil$ using the same speedup as above. However, the egress element now requires a buffer of size

NK, which can presumably be implemented on-chip. Further, it was shown that the same amount of buffers in the ingress element allows to emulate the OQ switch, within the same delay, using no speedup at all. It is important however to note that the emulation schemes need to maintain temporal lists that allow to determine the departure time.

Note that multi-path switches, in general, suffer from out-of-sequence arrivals at the egress element. This implies that local sequence control algorithms might be required. A summary of our findings on open-loop sequence control can be found in [43].

4.5 QoS in Multi-Module Switches

We now propose a two-phase algorithm[6] called *switched fair airport*, which enables QoS provisioning in a multi-module switch. The idea is to use proven f_1 and f_4 equivalences of a switch to provide isolated bandwidth guarantees and fairness in sharing excess bandwidth, respectively, in a fashion similar to an OQ switch, without the need for exact emulation (f_5). Our reference OQ switch employs the fair airport scheduler at each output link. A guaranteed-QoS (see Appendix A for definition) flow f belonging to the input-output pair (i, j) is associated with a rate $r_{i,j}^{(f)}$, which is scheduled using a non-work conserving policy (e.g., a shaped virtual clock). Each flow, including best-effort flows, is also associated with a weight $w_{i,j}^{(f)}$, which is used by a work conserving scheduler (e.g., weighted round robin) to fill up the slots unused by the guaranteed portion. We assume a finite buffer size B associated with each output j, which uses a fair buffer management scheme (which also guarantees buffer allocations).

The switched fair airport algorithm is a combination of two phases, the first that ensures f_1 equivalence with an OQ switch for a shaped portion of the flows, and the second that ensures f_4 equivalence. We will illustrate its working on a CIOQ switch, and then describe its extension to other single-path Buffered Clos switches. Consider an $N \times N$ CIOQ switch, in which the output elements employ the *same* per-flow fair airport scheduler as in the reference OQ switch. An input element i contains a non-work conserving scheduler per output j, which serves the k flows for the pair (i, j) by shaping the flow traffic into a guaranteed virtual output queue (G-VOQ) $Q_{i,j}^{(g)}$. For implementation purposes, these queues contain only the occupancy counters for each flow, and the cells continue to reside in the k per-flow queues associated with the pair (i, j). The total average bandwidth into the G-VOQ is bound as

$$\lambda_{i,j}^{(g)} \leq \sum_{f \in (i,j)} r_{i,j}^{(f)}$$

In the first phase, a maximal size matching, with speedup 2, is used to match on the basis of $Q_{i,j}^{(g)}$. Due to the admissibility of $\lambda_{i,j}^{(g)}$, over all input-output pairs, this step essentially provides a virtual bandwidth trunk of the aggregated rate allocations for each pair (i, j). The shaping within each pair ensures that the trunk bandwidth is properly divided among the k flows in the pair[7].

Conjecture 5 *Rate shaping of flows at every input-output pair, followed by a maximal size matching is sufficient to ensure the individual flow rates through the space element of a CIOQ.*

[6]We call this scheme switched fair airport due to its resemblance with the fair airport algorithm [27] for OQ.

[7]We have recently proved this conjecture to be true, using the Clos theorem (surprise, surprise!). However, we shall call it a conjecture, pending its publication.

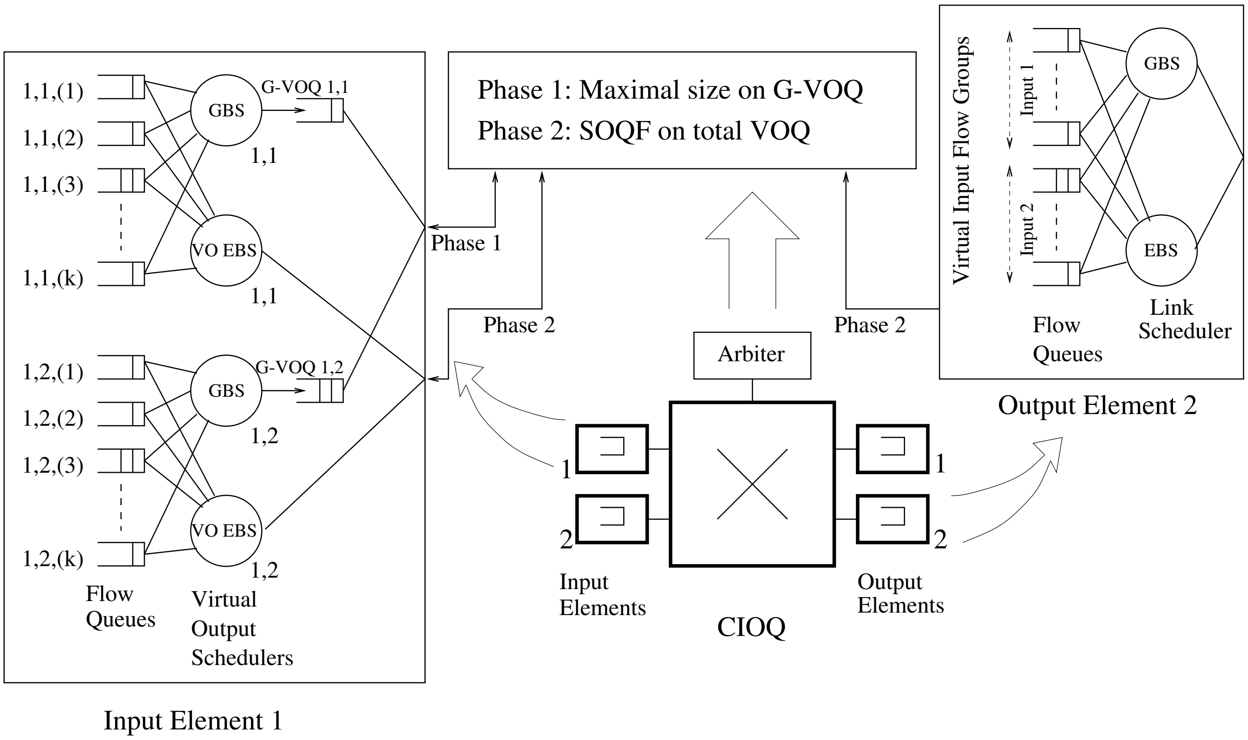


Figure 4.8: QoS in CIOQ using Switched Fair Airport

In the second phase, the *same* maximal size matching adds connections using SOQF. For this purpose, an output element maintains virtual input queue occupancies $Q'_{i,j}$, and an input element contains a virtual output EBS scheduler, which when enabled by an added connection in the second phase, distributes the excess bandwidth according to the same weight $w_{i,j}^{(f)}$. The intuition behind this approach is that if an input-output pair remains stable (in the strictest sense) when its total arrival rate is $w_{i,j}$ times the available excess bandwidth, then a flow within that pair remains stable when its arrival rate is $w_{i,j}^{(f)}$ times the available bandwidth on the output link. However, this still remains a conjecture as we have not thoroughly analyzed the scheme yet.

Conjecture 6 *The SOQF algorithm, combined with a virtual-output weight-scheduler, maintains the same per-flow weighted allocation of the output link excess bandwidth as an OQ switch.*

Note that SOQF provides the function of feedback control from the output link scheduler by giving priority, in the matching, to flows whose service rate at the input element does not overstep the service rate at the output element. Note also that the effect of finite buffers is not completely clear. We assume that a buffer of size B (same as in each output of a memory element) is now present in each of the input and output elements with the same buffer management scheme to control the residues of scheduling.

The two-phase switched fair airport queueing structure is illustrated in Fig. 4.8. This approach can be easily extended to CIOQ-A that shadows a CIOQ by maintaining the same queueing structure as above. Similarly, it may be extended to CIOQ-P and G-MSM switches that emulate a CIOQ, and to equal dispatch based systems, by using the same rates and weights on each path for an input-output pair. (An elaboration of these extensions is still in progress.)

Providing QoS guarantees for the PPS architecture is more involved. Bandwidth guarantees may be provided by using a flow-based PPS. However, the throughput as well as the fairness in the allocation of excess bandwidth is not comparable to an OQ switch. Presumably, a packet-by-packet PPS may be used with equal or fractional dispatch on a *per flow* basis, with each core element containing the same output link scheduler as the reference switch, with the same weights for the subflows (since each core element receives no more than $1/K$ of the flow traffic). However, this also needs further investigation.

Chapter 5

Conclusions

We addressed the problem of building a high capacity QoS-capable packet switch using stages of lower capacity memory and space elements. Towards that end we proposed a taxonomy of fully connected three-stage switches called Buffered Clos switches, and provided a formal framework to study their performance. Within this framework, we presented algorithms that provide functional equivalence with an ideal reference switch for the aggregated and pipelined CIOQ, general memory-space-memory switches, and the parallel packet switch architecture. To apply the QoS methodologies to multi-module switches, we introduced the switched fair airport algorithm.

The following are some items that need to be addressed. The SOQF algorithm needs to be analyzed, and the related conjectures proven. The work on PPS needs further investigation to bound the mis-sequencing and egress buffers for the fractional dispatch algorithm. Also, the work on switched fair airport is fairly nascent and needs to be concluded.

We would like to thank Denis Khotimsky and Dimitrios Stiliadis for several productive technical discussions on this subject, which allowed to crystallize some of the ideas presented here.

Appendix A

Switching Primer

A.1 Properties of Circuit Switches

Circuit switched networks rely on dividing the physical communication media into synchronous units called *channels*. For example, in time division multiplexed (TDM) networks, the time axis of a transmission link is divided into timeslots, while in wavelength division multiplexed (WDM) and frequency division multiplexed (FDM) networks, a channel corresponds to a carrier wavelength or frequency, respectively. A *circuit* is established by pre-determining the path to be traversed through the network, and assigning channels, if they are available, on the links that comprise the path. The allowed circuit bit rates, the *frame* sizes to be transmitted in each synchronous unit, and correspondingly the size of the unit, are governed by digital transmission standards, such as SDH and SONET [16] for TDM networks. A circuit switched node transfers the frames that arrive on a given channel of an input link to a pre-configured channel of an output link. Examples of such systems include SONET switches that employ electronic crossbars, and more recently, wavelength routers based on optical components such as tunable lasers and Micro-Electro-Mechanical (MEMS) mirrors.

A *contention* occurs when more than one frame arrive at a given link at the same instant. Since channels are allocated on every network link of the end-to-end path of a circuit, prior to frame transmission, there is no *external* contention for the output link resources of a circuit switch, and hence no inherent necessity for buffers[1] to absorb contention. Consequently, the performance measures of interest are related only to the admissibility of circuits, also called *call-level behavior*, and the realization of internal paths, through the switching node, for the admitted circuits. The former is studied using so called *loss models* [61], which characterize circuit acceptance probabilities, based on the given statistics of call arrivals and holding times. A celebrated example is the Erlang loss system which has been used for years to engineer telephony networks. The issue of path realization, on the other hand, deals with the architecture of the switching node itself. By definition, a node achieves maximum throughput as long as a switching path can be established for every circuit that is admissible solely on the basis of the available bandwidth of the external links. Once a circuit is admitted and a path is realized, the frame-level behavior, in terms of the observed frame bandwidth and delays, is guaranteed without additional mechanisms.

A bipartite mapping from the input to the output links of a switch, during a given time unit, is

[1] A fixed amount of frame buffers may be used to convert from one channel to another, e.g., for interchanging timeslots, an operation called *grooming*.

referred to as a *matching* if each input is connected to no more than one output, and vice-versa. A switch is called *non-blocking* if internal paths can be set up to satisfy any given matching. It is said to be non-blocking in the strict sense if a path can be found between an idle input-output pair without disturbing the ones that support any existing matching. If a re-arrangement of the existing paths is necessary and sufficient to support the new pair, the switch is called re-arrangeably non-blocking. All non-blocking switches are *functionally equivalent* in terms of the matchings that they can realize. Note that while a matching determines the input-output pairs to be connected, path establishment within the switch determines the feasibility of the given matching. In circuit switched networks, both the sequence of matchings and the associated paths are determined at the call-level, prior to actual frame transmission.

A trivial example of a strictly non-blocking $N \times M$ switching fabric is a crossbar that employs $N \cdot M$ electronic crosspoints, configured using the current matching, in order to connect N inputs to M outputs. There is a unique path, between an input and an output, which is readily available when both are idle. Much of the work in circuit switching [29, 57] addresses the construction of a bufferless fabric using an inter-connection network of smaller components, motivated primarily by the fact that electronic crosspoints were expensive, and their number in a crossbar increases quadratically with the size of the switch. Such networks may have *internal* contention due to commonality in the internal paths between input-output pairs, and may employ internal *speedup* to counter blocking, which is defined as the ratio of the total link bandwidth between two stages, to the total external bandwidth. Examples of notable inter-connection networks are the Banyan network and the Batcher sorting network, which require $O(N \log N)$ and $O(N \log^2 N)$ crosspoints, respectively. Both these networks are *self-routing* with a unique path between an input-output pair, and hence do not require centralized path set-up, however, the resulting structure is blocking. A popular example of a non-blocking network, which we shall revisit on several occasions, is the 3-stage (and recursive) Clos network, which uses $O(N^{1.5})$ crosspoints. A more complex example is a Cantor network, which uses $O(N \log^2 N)$ crosspoints and remains non-blocking by employing several planes of a (blocking) sorting network. Indeed, the literature in the design of bufferless inter-connection networks is fairly rich, yet rigorous due to the well defined cost parameters and performance measures, i.e., the number of crosspoints and blocking behavior, respectively.

A.2 Packet Switch Design

Packet switched networks do not rely on holding dedicated link resources and on switching pre-established physical layer circuits. Instead, the physical and link layers have the flexibility to employ either synchronous means such as SONET, or asynchronous ones such as Ethernet, on a link-by-link basis. The forwarding decisions at each packet switched node, on the end-to-end path, are made on a per-packet basis. A packet may traverse through multiple nodes that are connected to each other through heterogeneous link layers, where a logical link may itself be comprised of a physical circuit that spans several circuit switched nodes. In connection-oriented networks such as ATM, the packet[2] header contains a virtual circuit (VC) identifier, which corresponds to a flow and is used to look up the path information, namely, the output interface to use for forwarding, as well as other service parameters, both of which are set up during a signaling phase prior to data transmission. Alternatively, in connectionless networks such as IP, the network layer address

[2]In ATM, this is actually referred to as a *cell*. However, we will continue to refer to all forwarded units as "packets", and reserve the term "cell" for the *internal* fixed size unit which is switched.

in the packet header is used to look up a routing table, which maintains reachability information using a concurrent control plane. Flows are identified by a header filter, which may be configured using a signaling phase (e.g., as in RSVP [74]) or through service level agreements (SLA), and may correspond to ranges of source and destination addresses, protocol types, and/or application port numbers. In either kind of network, a flow may be fine-grained, such as a single VoIP session configured via RSVP, or coarse-grained such as a permanent ATM VC providing connectivity between two IP subnets. The aggregate traffic between the input-output pairs of a switch may be considered as the flows with the coarsest granularity, as seen by that switch. Since dedicated resources need not be held for the entire lifetime of a flow, packet switched networks, in theory, provide better utilization due to statistical multiplexing of the link resources.

The foremost distinguishing feature of a packet switch is the inherent presence of external contention for the output link, i.e., several packets destined to the same output may arrive simultaneously. This phenomenon can be sustained over an arbitrary period of time, resulting in a backlog of unserved packets inside the switch, which needs to be *buffered*. An arriving packet may be dropped in response to congestion in the finite amount of available buffers, yielding a packet loss ratio for the flow. The admitted backlog is then *scheduled* in a chosen fashion, which determines the packet delays and the observed flow service rates.

The traffic presented to a typical packet switch consists of a combination of *guaranteed QoS* flows, for which the traffic profiles and service requirements, such as desired average bandwidth and packet delay bound, are known in advance, and *best effort* flows without any pre-specified profile or requirements. Guaranteed QoS flows undergo a call admission control (CAC) procedure during the signaling phase. This procedure uses the profile (e.g., a leaky bucket specification [18]) of the flow and the service requirements, in order to ascertain if the switch has enough resources, in the longer term, to meet those specifications. If successful, *effective* bandwidth and buffer size values are calculated, which are then used to program the scheduling and buffer management schemes within the switch so that the agreed upon requirements may be honored. In addition, policers at the inputs of the switch ensure that the traffic sources adhere to the advertised profiles. Often, the flows are allowed to violate their profiles, however, the excess component of the traffic is treated on a best effort basis. While loss models [61], similar to the ones in circuit switching, may be used to study the call-level behavior of the guaranteed QoS flows, and hence to engineer the network, it is ultimately the scheduling and buffer management policies, i.e., the *packet-level behavior*, that govern the feasibility of the longer term call admission, and thereby the QoS capability of the chosen switch. On the other hand, best effort flows, which comprise the majority of the Internet traffic today, do not undergo similar CAC procedures or conventional loss models for network engineering. Instead, empirical studies [58, 4] are used to characterize the statistical nature of the flows in the chosen network segments, which then yield the expected amount of effective bandwidth and buffers required to accommodate them in the network nodes, and thereby to engineer the network. Consequently, the performance measures of interest for the best effort traffic are related mainly to optimality in packet throughput.

In summary, the primary goals in the design of a packet switched node is to sustain optimal throughput (given a definition of optimality) and to provide preferential treatment, in the form of scheduling and buffer management, to specified flows, in the presence of external and (possibly) internal contention.

Appendix B

Scheduling and Buffer Management

B.1 Link Scheduling

Consider an OQ switch, composed of a single $N \times N$ memory element. To enable bandwidth, delay, and fairness guarantees, packets are enqueued into separate *per-flow* queues at each output, and are dequeued using a chosen scheduling policy. A flow may correspond to a single end-to-end connection, or to aggregates of several (including best-effort) connections, depending upon the level at which the scheduler operates. A simple instance of a scheduler is the Weighted Round Robin (WRR) [38] poller, in which each flow is guaranteed a bandwidth that is proportional to its weight, and experiences a worst-case latency equal to the size of the WRR frame. Guaranteed bandwidth as well as fairness are linked to the same weight. A popular variant which accounts for variable packet sizes is the Deficit Round Robin [62] policy.

A classical rate-based scheme, which provided a framework for perfect isolation, is the fluid-model Generalized Processor Sharing (GPS) [56] scheme. Each flow i is associated with a rate r_i, and receives an instantaneous service rate proportional to that rate, whenever it is backlogged. The scheduler operates in a work conserving fashion, and consequently, the flow rate is guaranteed as long as the sum of the rates is less than the link capacity C. Due to the rate proportional service model, if two flows i and j are continuously backlogged in an interval (t_1, t_2), the amount of service received by the flows are related as

$$W_i(t_1, t_2)/r_i = W_j(t_1, t_2)/r_j$$

The GPS scheduler has several interesting properties. Firstly, as seen from the above equation, the *normalized* service received by each flow is equal at every instant. Therefore, the excess bandwidth is also distributed in proportion to the flow rates, leading to one of the first formal definitions of *service fairness* as the worst-case difference (between any two backlogged flows) in normalized service received. Secondly, if $B(t)$ is the set of backlogged flows at time t, the instantaneous service rate observed by a non-empty queue i is equal to $r_i \cdot C/\sum_{j \in B(t)} r_j$. That is, the flow receives isolated bandwidth at every instant, or with zero latency. Thirdly, the observed delay depends only upon the flow's own arrival process, as shown in B.1. For example, if the flow i is leaky bucket constrained [18] with a bucket size of σ_i, the delay is bounded by σ_i/r_i.

The packetized version of GPS, called Weighted Fair Queueing (WFQ) [20], works by essentially simulating a fluid system by keeping track of the normalized work done, also called *virtual time* $v(t)$ at every timeslot. Note that, in GPS, $v'(t) = C/\sum_{i \in B(t)} r_i$. On arrival, the kth packet of

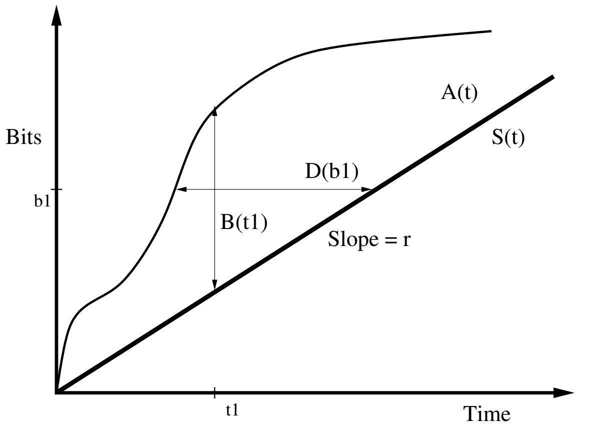


Figure B.1: GPS: Delays and Backlog

flow i with length l_i^k is time-stamped using its finishing virtual time F_i^k, which is computed as:

$$F_i^k = l_i^k / r_i + \max(v(t), F_i^{k-1})$$

This policy simulates GPS with an added latency of $(L_i/r_i + L_{\max}/C)$. Several of the subsequent packet-based scheduling algorithms proposed the use of approximations of $v(t)$ that were easier to compute, but yielded different delay bounds and service fairness. For example, in Virtual Clock [73], $v(t)$ is set to real time t, which leads to delay bounds comparable to WFQ but unbounded service fairness. In Self-Clocked Fair Queueing (SCFQ) [26], the virtual time is set to the timestamp of the currently served packet, yielding the best fairness properties among GPS-related schedulers, but a delay bound which is $O(n)$, where n is the number of flows. A general model for packet-based schedulers, called rate proportional [64] servers, was later devised showing that any scheme in which $v(t)$ grows atleast as fast as real time, but always under-estimates it with respect to a GPS server, has the same added latency as WFQ, with the service fairness dependent only on the extent of under-estimation. Examples of such schemes include the Virtual Clock and Frame-Based Fair Queueing (FFQ) [65].

The notion of *worst-case* fairness was introduced in [2], especially suited for hierarchical [3] (see Fig. B.2(a)) and distributed schedulers, and was defined as the worst-case inter-service time. It was shown that the added latency observed by a packet is equal to the sum of the worst-case fairness indices of the intermediate nodes. In contrast to all the above schemes in which fairness is tied to the associated rate, the Fair Airport [27] algorithm allowed to decouple the scheduler into a non-work-conserving guaranteed bandwidth scheduler (GBS) (e.g., shaped Virtual Clock) to ensure rates and delays, and a work conserving excess bandwidth scheduler (EBS) (e.g., WRR) as shown in Fig. B.2(b). For the purposes of this work, we will assume the existence of an implementable [9] link scheduling scheme which provides a method to isolate bandwidth on a specified time-scale, and to distribute excess bandwidth according to pre-specified weights. The most recent work in link scheduling is primarily focused on the statistical characterization of GPS-based and Earliest Deadline First (EDF) schedulers, as well as more complicated service curves, and is beyond the scope of this work.

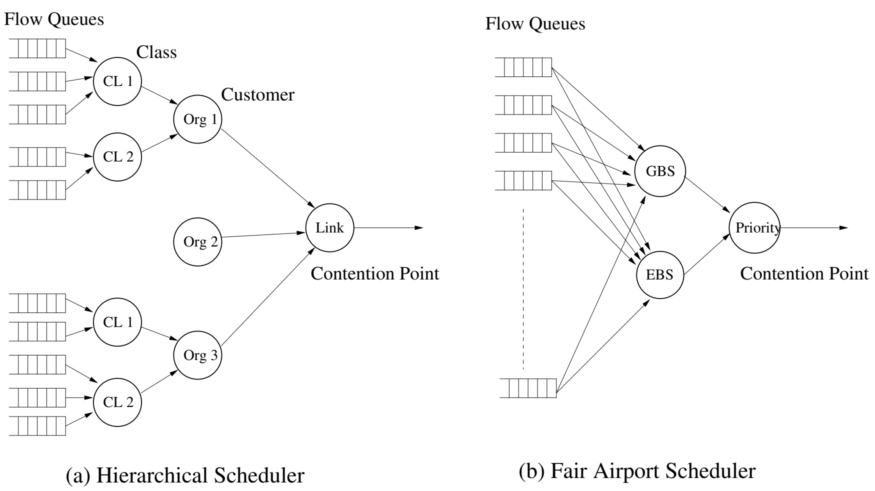


Figure B.2: Scheduler Arrangements: (a) Hierarchical (b) Fair Airport

B.2 Buffer Management

The work on scheduling and arbitration algorithms focus on the finite bandwidth resource at a contention point, assuming essentially an infinite attached memory so that bandwidth isolation, inter-queue fairness and overall throughput depends solely on those algorithms. Given a finite buffer resource in a memory element, we may view buffer management as a mechanism to control the residue of scheduling to enable *unhindered* performance of the above bandwidth schedulers, by selectively admitting packets into the finite memory. Therefore, we identify three, often conflicting goals: (i) to provide isolation to guaranteed QoS flows; (ii) to allow fair access to the available buffers, in order to support fair scheduling; and (iii) to maintain high throughput by sharing the available buffers among the served flows. Much like the literature on scheduling, the primary focus has been on OQ switches, and the application to other switch designs is still immature.

The notion of isolation, in the context of buffers, is well defined. Given a traffic profile, which is readily available for guaranteed QoS flows, and a service rate, a buffer size b_i may be calculated for either zero loss for regulated traffic (e.g., σ_i plus the scheduler latency for a leaky-bucket regulated source served by a virtual bandwidth trunk with service rate equal to the token rate) or a specified loss ratio for statistical traffic (e.g., using the tail of the distribution of a queue). Once the size is computed, it can be totally isolated by using the *complete partitioning* [32] scheme, in which b_i is dedicated to flow i with $\sum_i b_i \leq B$, the total buffer size. A packet which causes a queue length to exceed this size is dropped on arrival. This is a special case of *static thresholding* [24, 47], in which the sum of sizes may be allowed to exceed the total buffer size. While the latter does not provide perfect isolation, it allows to share the buffers (in a limited way) by utilizing a knowledge of the inter-dependencies of arrival processes.

While the concept of fairness is well established for link schedulers, the same cannot be said for fair allocation of buffers to store the residue of the excess-bandwidth traffic served by a scheduler, due to a lack of arrival traffic characterization. Consequently, an *equal* access to the buffers is often considered fair, and may be achieved by static thresholding. However, the throughput decreases with respect to the case when each flow has full access to the available memory. The *dynamic thresholding* scheme [15] aims to equalize the queue lengths, while increasing the throughput by dynamically changing the allocation, so that the total queue length may always approach the total buffer size. The threshold is calculated as α times the current free space, and weighted access may be offered by choosing different values of α.

In contrast to the above schemes, all of which are *arrival-drop* policies, the *push-out* [25] scheme allows to maintain full buffer occupancy, and hence the highest throughput for a given B, with exactly equal allocations to each backlogged flow. This is achieved by expelling an already enqueued packet from the queue with the longest length. The choice of the packet within the longest queue is flexible, e.g., in [67], it is suggested that dropping the head packet is the most beneficial for TCP flows. In fact, dynamic thresholding may be considered as an arrival-drop approximation of push-out. Another attractive property of push-out schemes is that even the portion of memory providing isolated allocations may be completely shared, by slightly modifying the push-out criterion. When the buffer is full, a packet is expelled from a queue, whose length is the greatest over a specified threshold. Inspite of such properties, push-out is difficult to implement at wire speeds, since it requires the maintenance of sorted queue lengths and an extra memory access. The *dynamic partitioning* scheme [45], can indeed be considered as an arrival-drop approximation of push-out with thresholds.

There are several schemes that enforce intra-queue fairness, in which a single FIFO queue is composed of packets belonging to several best-effort flows. By essentially regulating the entry of packets belonging to different flows into the same queue, such schemes also may be viewed as single-FIFO scheduling schemes as they control the fraction of the total queue bandwidth that each flow receives. Popular examples are Random Early Detection (RED) [23] and its several subsequent variants, as well as the recent AQM schemes. In our framework, if several flows share the same queue, we assume that some such algorithm is also present, in addition to the inter-queue schemes of our interest, and we shall not elaborate on their effects.

Appendix C

Proofs of Theorems

C.1 Single-Path Switches

Theorem 1

$$(\text{CIOQ-A}, \{\text{Maximal Size}\}) \overset{T,f_2}{\simeq} (\text{OQ}, \{\text{WC}\}), \quad s = 2$$

Proof: Let i, j be an input and output element (not interface), respectively. Let $\lambda_{N\times N}$ be the rate matrix of the switch. Since, for f_2, we assume that the rates are admissible, we know that

$$\forall i \sum_j \lambda_{i,j} < 1, \ \ \forall j \sum_i \lambda_{i,j} < 1.$$

Compose an aggregate rate matrix $\lambda^*_{P\times P}$, which adds up the rates on each input and output element as follows:

$$\forall i, j \quad \lambda^*_{i,j} = \sum_{k=\frac{N}{P}(i-1)+1}^{k\leq\frac{N}{P}i} \ \sum_{l=\frac{N}{P}(j-1)+1}^{l\leq\frac{N}{P}j} \lambda_{k,l}$$

Due to the admissibility of the original rate matrix, and the summation above, we can easily see that

$$\forall i \sum_j \lambda^*_{i,j} < N/P, \ \ \forall j \sum_i \lambda^*_{i,j} < N/P.$$

If we re-normalize time so that each timeslot now is $\frac{1}{N/P}$ times the external timeslot, then the rates become admissible for the smaller timeslot (faster time). Consequently, according to [19], a maximal size matching algorithm with a speedup of 2 is sufficient to ensure the stability of the queues in the system. Since the space element operates at a frequency of sN/P, it follows that $s = 2$ is sufficient for functional equivalence at level f_2.

Theorem 4

$$(\text{CIOQ-P}, \{\text{Equal Dispatch, Maximal Size}\}) \overset{T,f_2}{\simeq} (\text{OQ}, \{\text{WC}\}), \quad s = 2$$

Proof: Let i, j be an input and output interface, respectively. Let $\lambda_{N\times N}$ be the rate matrix of the switch, whose admissibility leads to

$$\forall i \sum_j \lambda_{i,j} < 1, \ \ \forall j \sum_i \lambda_{i,j} < 1.$$

Let $\lambda_{i,j}^{(k)}$ be the average arrival rate into the per-path VOQ $Q_{i,j}^{(k)}$. Since an input-output traffic flow is distributed equally among all the per-path VOQs for that pair, we have

$$\forall k \left\{ \forall i \sum_j \lambda_{i,j}^{(k)} < 1/K, \;\; \forall j \sum_i \lambda_{i,j}^{(k)} < 1/K. \right\}$$

We re-normalize time so that each timeslot is now K times the external timeslot. Due to the above inequality, the rates continue to remain admissible, and a maximal size matching algorithm [19] with a speedup of 2 on the larger timeslot is sufficient to ensure the stability of each of the VOQs for each of the path. Since each space element operates at a frequency of s/K, it follows that $s = 2$ is sufficient to ensure relative stability with an OQ switch for admissible rates.

C.2 Multi-path Switches

Theorem 9

$$\text{(PPS, \{Fractional Dispatch\})} \stackrel{T,f_3}{\simeq} \text{(OQ, \{WC\})}, \quad X = [MMM], s \geq \frac{\lceil R/r \rceil}{R/r}$$

Proof: By the definition of the fractional dispatch algorithm, the average rate for an input-output pair (i, j) in core element k is given by:

$$\lambda_{i,j}^{(k)} \leq \frac{1}{\lceil \frac{R}{r} \rceil} \lambda_{i,j}$$

Consequently, the total traffic destined to an output j, which is stable in a reference OQ switch can be calculated as follows:

$$\begin{aligned} \sum_i \lambda_{i,j}^{(k)} &\leq \frac{1}{\lceil \frac{R}{r} \rceil} \sum_i \lambda_{i,j} \\ &< \frac{R}{\lceil \frac{R}{r} \rceil} \leq r \end{aligned}$$

Therefore, the output j in each core element k is also stable, thereby establishing relative stability in the strict sense. Note that the proof continues to hold when the traffic to the input-output pair (i, j) is less than $w_{i,j}R$, in which case the same for each core element becomes less than $w_{i,j}r$. In other words, the fractional dispatch algorithm is also sufficient for relative stability in the strictest sense.

Bibliography

[1] T. Anderson, S. Owicki, J. Saxe, C. Thacker, “High speed Switch Scheduling for Local Area Networks,” *ACM Trans. Computer Systems*, Nov 1993.

[2] J. C. R. Bennett and H. Zhang, “WF2Q: Worst-Case Fair Weighted Fair Queueing,” in *Proc. IEEE Infocom ‘96*, pp. 120-128, Mar 1996.

[3] J. C. R. Bennett and H. Zhang, “Hierarchical Packet Fair Queueing Algorithms,” in *Proc. ACM Sigcomm ‘96*, pp. 143-156, Aug 1996.

[4] J. Cao, W. S. Cleveland, D. Lin, D. X. Sun, “On the Nonstationarity of Internet Traffic,” in *Proc. ACM Sigmetrics*, Cambridge, MA, Jun 2001.

[5] C. S. Chang, J. W. Chen, H. Y. Huang, “On Service Guarantees for Input-Buffered crossbar switches: A capacity decomposition approach by Birkhoff and Von-Neumann,” in *Proc. IEEE Intl. Workshop on QoS*, London 1999.

[6] C. S. Chang, J. W. Chen, H. Y. Huang, “Birkhoff-Von Neumann Input-Buffered Crossbar Switches,” in *Proc. IEEE Infocom ‘00*, Tel Aviv, pp. 1614-1623, Mar 2000.

[7] F. M. Chiussi, “Design, Performance and Implementation of a Three-Stage Banyan-based Architecture with Input and Output buffers for Large Fast Packet Switches,” *PhD. Thesis*, Stanford University, Jul 1993.

[8] F. M. Chiussi and A. Francini, “Providing QoS Guarantees in a Packet Switch,” in *Proc. IEEE Globecom ‘99*, Nov 1999.

[9] F. M. Chiussi, A. Francini, J. G. Kneuer, “Implementing Fair Queueing in ATM Switches,” (Parts 1-2) in *Proc. IEEE Globecom ‘97*, Phoenix, AZ, Nov 1997.

[10] F. M. Chiussi, D. A. Khotimsky, S. Krishnan, “Generalized Inverse Multiplexing of Switched ATM Connections,” in *Proc. Globecom ‘98*, Sydney, Nov 1998.

[11] F. M. Chiussi, D. A. Khotimsky, S. Krishnan, “Advanced Frame Recovery in Switched Connections Inverse Multiplexing for ATM,” in *Proc. Intl. Conf. on ATM*, Colmar, France, Jun 1999.

[12] F. M. Chiussi, J. G. Kneuer, V. P. Kumar, “Low-cost Scalable switching solutions for Broadband Networking: the ATLANTA architecture and chipset,” *IEEE Communications Magazine*, vol. 35, no. 12, Dec 1997.

[13] F. M. Chiussi, *et al.*, "A Chipset for Scalable QoS-Preserving Protocol-Independent Packet Switch Fabrics," in *Proc. Intl. Solid-State Circuits Conf.*, San Francisco, Feb 2001.

[14] F. M. Chiussi, *et al.*, "A Family of ASIC devices for Next Generation Distributed Packet Switches with QoS support for IP and ATM," in *Proc. Hot Interconnects 9*, Stanford, Aug 2001.

[15] A. K. Choudhury and E. L. Hahne, "Dynamic Queue Length Thresholds in a Shared Memory ATM Switch," in *Proc. IEEE Infocom '96*, pp. 679-687, Mar 1996.

[16] M-C. Chow, "Understanding SONET/SDH: Standards and Applications," Adnan Publisher, New Jersey, 1995.

[17] S-T. Chuang, A. Goel, N. McKeown, B. Prabhakar, "Matching Output Queueing with a Combined Input Output Queued Switch," *IEEE J. Selected Areas of Communications*, vol. 17, no. 6, Jun 1999.

[18] R. L. Cruz, "A Calculus for Network Delay, Part I: Network Elements in Isolation," *IEEE Trans. Information Theory*, vol. 37, no. 1, Jan 1991.

[19] J. G. Dai and B. Prabhakar, "The throughput of data switches with and without speedup," in *Proc. Infocom 2000*, Tel Aviv, Mar 2000.

[20] A. Demers, S. Keshav, S. Shenker, "Analysis and Simulation of a Fair Queueing Algorithm," in *Proc. Sigcomm '89*, Austin, TX, Sep 1989.

[21] M. Devault, J. Cochennec, M. Servel, "The Prelude ATD Experiment: Assessments and Future Prospects," *IEEE J. Selected Areas in Comm.*, vol. 6, no. 9, Dec 1988.

[22] A. Elwalid, D. Mitra, R. Wentworth, "A New Approach for Allocating Buffers and Bandwidth to Heterogenous, Regulated Traffic in an ATM node," *IEEE J. Selected Areas of Communications*, vol. 13, pp. 1115-1127, Aug 1995.

[23] S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," *IEEE Transactions on Networking*, Aug 1993.

[24] G. J. Foschini and B. Gopinath, "Sharing Memory Optimally," *IEEE Transactions on Communications*, vol. 3, pp. 352-360, Mar 1983.

[25] L. Georgiadis, I. Cidon, R. Geurin, A. Khamisy, "Optimal Buffer Sharing," *IEEE J. Selected Areas of Communications*, vol. 13, pp. 1229-1240, Sep 1995.

[26] S. J. Golestani, "A Self-Clocked Fair Queueing scheme for Broadband Applications," in *Proc. IEEE Infocom '94*, Jun 1994.

[27] P. Goyal and H. M. Vin, "Fair Airport Scheduling Algorithm," in *Proc. NOSSDAV '97*, pp. 272-283, May 1997.

[28] J. E. Hopcroft and R. M. Karp, "An $n^{5/2}$ algorithm for maximum matching in bipartite graphs," *SIAM Journal of Computation*, vol. 2, 1973.

[29] J. Y. Hui, "Switching and Traffic Theory for Integrated Broadband Networks," Kluwer Academic Press, 1990.

[30] A. Hung, G. Kesidis, N. McKeown, "ATM Input-Buffered switches with Guaranteed Rate property," in *Proc. IEEE ISCC*, Athens, 1998.

[31] I. Iliadis and W. E. Denzel, "Performance of packet switches with input and output queueing." in *Proc. ICC '90*, Atlanta, Apr 1990.

[32] M. I. Irland, "Buffer Management in a Packet Switch," *IEEE Trans. Communications*, vol. 26, pp. 328-337, Mar 1978.

[33] S. Iyer, A. Awadallah, N. McKeown, "Analysis of a Packet Switch with Memories running slower than line rate," in *Proc. Infocom 2000*, Tel Aviv, Mar 2000.

[34] S. Iyer and N. McKeown, "Making Parallel Packet Switches practical," in *Proc. Infocom '01*, Anchorage, Alaska, Apr 2001.

[35] S. Iyer, R. Zhang, N. McKeown, "Routers with a Single Stage of Buffering," to appear in *ACM Sigcomm '02*, Pittsburgh, Sep 2002.

[36] K. Kar, T. V. Lakshman, D. Stiliadis, L. Tassiulas, "Reduced Complexity Input-Buffered Switches," in *Proc. Hot Interconnects 8*, Palo Alto, Aug 2000.

[37] M. Karol, M. Hluchyj, S. Morgan, "Input versus Output queueing on a Space Division Switch," *IEEE Trans. Communications*, vol. 35, no. 12, Dec 1987.

[38] M. Katevenis, S. Sidoropoulus, C. Courcoubetis, "Weighted Round Robin Cell Multiplexing in a General Purpose ATM Switch chip," *IEEE J. Selected Areas in Communications*, vol. 9, pp. 1265-1279, Oct 1991.

[39] I. Keslassy, M. Kodialam, T. V. Lakshman, D. Stiliadis, "On Guaranteed Smooth Scheduling for Input-Queued Switches," submitted to *IEEE Infocom '03*, 2003.

[40] D. A. Khotimsky, "A Packet Re-sequencing Protocol for Fault-Tolerant Multi-Path transmission with Non-Uniform Traffic Splitting," in *Proc. Globecom '99*, Rio de Janeiro, Dec 1999.

[41] D. A. Khotimsky and S. Krishnan, "Towards the Recognition of Parallel Packet Switches," *Gigabit Networking Workshop at Infocom '01*, Anchorage, Alaska, Apr 2001.

[42] D. A. Khotimsky and S. Krishnan, "Stability Analysis of a Parallel Packet Switch with Bufferless input Demultiplexors," in *Proc. ICC '01*, Helsinki, Jun 2001.

[43] D. A. Khotimsky and S. Krishnan, "Evaluation of Open-loop Sequence Control schemes for Multi-path Switches," in *Proc. ICC '02*, New York, Apr 2001.

[44] P. Krishna, N. S. Patel, A. Charny, R. Simcoe, "On the speedup required for work-conserving Crossbar Switches," in *Proc. 6th Intl. Workshop on QoS*, Napa, CA, May 1998.

[45] S. Krishnan, A. K. Choudhury, F. M. Chiussi, "Dynamic Partitioning: A Mechanism for Shared-Memory Management," in *Proc. IEEE Infocom '99*, pp. 144-152, New York, Mar 1996.

[46] P. R. Kumar and S. P. Meyn, "Stability of queueing networks and scheduling policies," *IEEE Transactions on Automatic Control*, vol. 40, no. 2, Feb 1995.

[47] G. Latouche, "Exponential Servers Sharing a Finite Storage: Comparison of Space Allocation Policies," *IEEE Transactions on Communications*, vol. 28, pp. 992-1003, Jun 1980.

[48] E. Leonardi, M. Mellia, F. Neri, M. A. Marsan, "On the Stability of Input-Queued switches with Speed-up," *IEEE/ACM Trans. Networking*, vol. 19, no. 1, Feb 2001.

[49] Y. Li, S. Panwar, H. J. Chao, "On the Performance of a Dual Round-Robin Switch," in *Proc. IEEE Infocom '01*, San Francisco, Apr 2001.

[50] N. McKeown, "Scheduling Algorithms for Input-Queued Cell Switches," *PhD. Thesis*, University of California at Berkeley, 1995.

[51] N. McKeown, "The iSLIP Scheduling Algorithm for Input-Queued Switch," *IEEE/ACM Transactions on Networking*, vol. 7, Apr 1999.

[52] N. McKeown, V. Anantharan, J. Walrand, "Achieving 100% Throughput in an Input-Queued Switch," in *Proc. Infocom '96*, San Francisco, Mar 1996.

[53] N. McKeown, A. Mekkittikul, "A Practical Scheduling Algorithm to achieve 100% throughput in Input-Queued Switches," in *Proc. IEEE Infocom '98*, San Francisco, Apr 1998.

[54] S. Melen and J. Tuner, "Non-blocking Multirate Networks," *SIAM Journal of Computing*, vol. 18, 1989.

[55] S. P. Meyn and R. Tweedie, "Markov Chains and Stochastic Stability," Springler-Verlag, London, 1994.

[56] A. K. Parekh and R. G. Gallagher, "A Generalized Processor Sharing approach to Flow Control in Integrated Service Networks– The Single Node Case," *IEEE/ACM Transactions on Networking*, Jun 1993.

[57] A. Pattavina, "Switching Theory: Architecture and Performance in Broadband ATM Networks," John Wiley, W. Sussex, UK, 1998.

[58] V. Paxson and S. Floyd, "Wide-Area Traffic: The failure of Poisson Modeling," in *Proc. ACM Sigcomm '94*, Aug 1994.

[59] B. Prabhakar, *Personal Communication*, Aug 2002.

[60] E. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol Label Switching Architecture," *Internet Engg. Task Force* RFC 3031, Jan 2001.

[61] K. W. Ross, "Multiservice Loss Models for Broadband Telecommunication Networks," Springer-Verlag, London, 1995.

[62] M. Shreedhar and G. Varghese, "Efficient Fair Queueing using Deficit Round Robin," in *Proc. ACM Sigcomm '95*, pp. 231-242, Sep 1995.

[63] D. C. Stephens and H. Zhang, "Implementing Distributed Packet Fair Queueing in a Scalable Switch Architecture," in *Proc. IEEE Infocom '98*, Mar 1998.

[64] D. Stiliadis and A. Varma, "A General Methodology for Designing Efficient Traffic Scheduling and Shaping Algorithms," in *Proc. IEEE Infocom '96*, Apr 1996.

[65] D. Stiliadis and A. Varma, "Design and Analysis of Frame-based Fair Queueing: A new traffic scheduling algorithm for packet switched networks," in *Proc. ACM Sigmetrics '96*, May 1996.

[66] I. Stoica and H. Zhang, "Exact emulation of an Output Queueing switch by a Combined Input Output queueing switch," in *Proc. 6th Intl. Workshop on QoS*, Napa, CA, May 1998.

[67] B. Suter, T. V. Lakshman, D. Stiliadis, A. K. Choudhury, "Design Considerations for supporting TCP with Per-flow Queueing," in *Proc. IEEE Infocom '98*, Mar 1998.

[68] Y. Tamir and G. Frazier, "High Performance multiqueue buffers for VLSI communication switches," in *Proc. 15th Annual Symp. on Computer Arch.*, Jun 19888.

[69] F. A. Tobagi, "Fast Packet Switch Architectures for Broadband Integrated Services Digital Networks," in *Proc. of IEEE*, vol. 78, no. 1, Jan 1990.

[70] B. Towles and W. J. Dally, "Guaranteed Scheduling for Switches with Configuration Overhead," in *Proc. IEEE Infocom '02*, New York, Jun 2002.

[71] G. Wilfong, B. Mikkelsen, C. Doerr, M. Zirngibl, "WDM Cross-connect Architectures with reduced complexity," *J. Lightwave Technology*, pp. 1732–1741, Oct 1999.

[72] Y. S. Yeh, M. G. Hluchyj, A. S. Acampora, "The Knockout Switch: A simple modular architecture for High-performance Packet Switching," *IEEE J. Selected Areas in Comm.*, vol. 5, no. 8, Oct 1987.

[73] L. Zhang, "Virtual Clock: A New Traffic Control Algorithm for Packet Switching," *ACM Transactions on Computer Systems*, May 1991.

[74] L. Zhang, S. Deering, D. Estrin, S. Shenker, D. Zappala, "RSVP: A new Resource Reservation Protocol," *IEEE Network*, vol. 7, no. 7, Sep 1993.