On the Provision of Quality-of-Service Guarantees for Input Queued Switches

Ge Nong and Mounir Hamdi

ABSTRACT

While the Internet has quietly served as a research and education vehicle for more than two decades, the last few years have witnessed its tremendous growth and its great potential for providing a wide variety of services. As a result, input-queued switching architectures, because of their distinguished advantage in building scalable switches, are currently receiving a lot of attention from both academia and industry as an attractive alternative for developing future-generation ATM/IP switches/routers. However, the problem of designing scheduling algorithms with QoS guarantees for input-queued switches has always been known to be a very challenging problem. In this article we give an overview of recent efforts in designing scheduling algorithms capable of providing QoS guarantees for input-queued switches. These algorithms are classified under three categories: those based on slot time assignment, those based on maximal matching, and those based on stable matching. We also present some open problems on this topic as future research directions in this area.

INTRODUCTION

While the Internet has quietly served as a research and education vehicle for more than two decades, the last few years have witnessed its tremendous growth and its great potential for providing a wide variety of services. Recently, the Internet has been growing at a very high rate. The number of hosts on the Internet has doubled approximately every 56 weeks since 1989, and the number of Web servers has doubled at least every 23 weeks for the last three years. Because the Internet is growing at an exponential rate and as common access line speeds increase, the Internet requires a switching/routing capacity of many gigabits per second of aggregate traffic.

Coupled with the tremendous physical growth of the Internet is the diversity of the services it can provide. In particular, there is a great demand for the Internet to provide quality-of-service (QoS) guarantees for a wide range of applications. Hence, there is an urgent need for the design of scalable and high-speed switches/routers that can provide QoS guarantees.

However, traditional architectures of Internet routers prevent the routers from being designed under high-speed environments. Furthermore, existing routers are expensive, not capable of providing QoS guarantees, and limited in throughput compared to recent high-speed switches. As a solution, recently there has been a trend in building high-speed Internet routers on top of *fast* packet switches such as asynchronous transfer mode (ATM) switches [1], because of their scalability and QoS provision capabilities.

Queuing schemes and scheduling algorithms are the two main factors affecting switch scalability and achievable performances. Specifically, queuing schemes provide ways to buffer the incoming packets and are the main factor affecting switch scalabilities. On the other hand, scheduling algorithms guarantee predictable switches' performances(i.e., QoS guarantees including throughput, packet delay, and jitter).

This article gives an overview of recent research efforts conducted in the design of scalable high-speed switches capable of providing QoS guarantees to various traffic streams. In this article we concentrate on the research done on nonblocking switching fabrics; that is, only external conflicts can occur at the input or output ports of the switch. In particular, we discuss algorithms that resolve external conflicts occurring at the input or output ports instead of internal conflicts occurring *only* in the blocking switching fabric. (External conflicts occurring at an input or output port refer to the fact that more than one cell may need to be transmitted in a time slot to the same input or output, respectively.) In fact, most state-of-the-art switches use nonblocking switching fabrics. The task of the scheduling algorithm used in a switch, therefore, is to resolve the input and output ports conflicts whenever there are any.

The remaining parts of the article are organized as follows. A brief summary on the architectures of *fast* packet switches is given in the next section. We then give a detailed description of the VOQ/CIOQ switches. Compared to an output-queued switch, we investigate the difficulties in designing algorithms for used by a VOQ or CIOQ switch to provide QoS guarantees for offered traffic. Algorithms using different methods to alleviate these difficulties are then introduced. Finally, a conclusion and topics for future studies are given.

QUEUING STRATEGIES

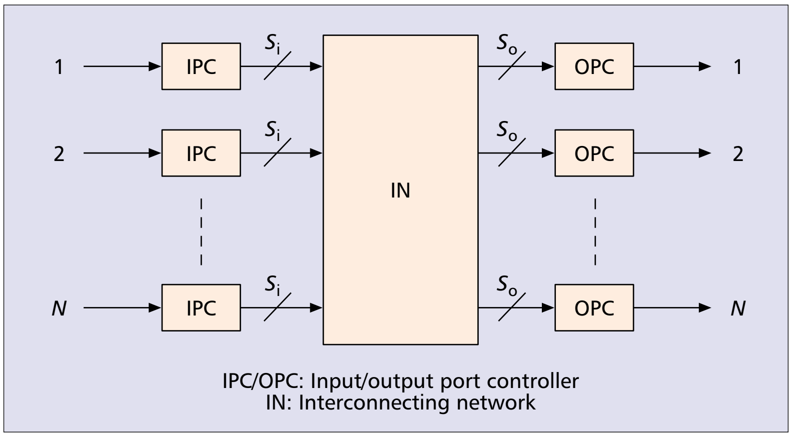
A general model of an $N \times N$ switch is shown in Fig. 1. The reference switch includes N input port controllers (IPCs), N output port controllers (OPCs), and an interconnecting network (IN). Each input/output link is assumed to transmit data signals at the same speed. Without loss of generality, the input/output link speed is supposed to be one packet per time slot. If the IN operates at a speed of S times each input/output link, it is said that the switch has an internal speedup of S. Therefore, in each time slot, an IN with internal speedup S is capable of switching up to S packets from each IPC and to each OPC, respectively. More specifically, in this article a switch with internal speedup S means that the switch performs scheduling and transmission of queued packets S times per time slot. In other words, a time slot is further split into Smini-slots, and each mini-slot is the time interval of performing one scheduling and transmission of queued packets.

Because of the unscheduled nature of packet arrivals to a switch, more than one packet may simultaneously arrive at different input ports and be destined for the same output port. With a speedup of one, the switch may allow only one of these contending packets to be immediately routed to the destined output port, but the others must be enqueued for transmissions thereafter. This form of congestion is unavoidable in a packet switch, and dealing with it often represents the greatest source of complexity in the switch architecture. A plethora of proposals for identifying suitable architectures for high-speed switches/routers have appeared in the literature [2]. These design proposals are based on various types of queuing strategies: output queuing (OQ), centralized shared queuing, input queuing, virtual output queuing (VOQ), or combined input-output queuing (CIOQ).

Output queuing: When a packet arrives at an input port, it is immediately put into the buffer that resides at the corresponding output port. Because packets destined for the same output port may arrive simultaneously from many input ports, the output buffer needs to enqueue traffic at a much higher rate (N times higher in the worst case, where N is the number of input ports) than a single port may dequeue it, which places stringent limits on switch size.

Centralized shared queuing: There is a single buffer shared by all the switch input ports, which can be viewed as a shared memory unit with N concurrent write accesses by the N input ports and up to N concurrent read accesses by the output ports. Because packets destined for the same output port may arrive simultaneously from many input ports, the output port needs to read traffic at a much higher rate than a single input port may write it, which places stringent limits on switch size.

Input queuing: Input queuing does not have the scaling limitations of OQ or centralized shared queuing. In this architecture, each input port maintains a first-in first-out (FIFO) queue of packets, and only the first packet in the queue is eligible for transmission during a given time slot. Regardless of its structure's simplicity, FIFO input-queued switches suffer from a performance bottleneck, namely head-of-line (HOL) blocking, which will be explained later.



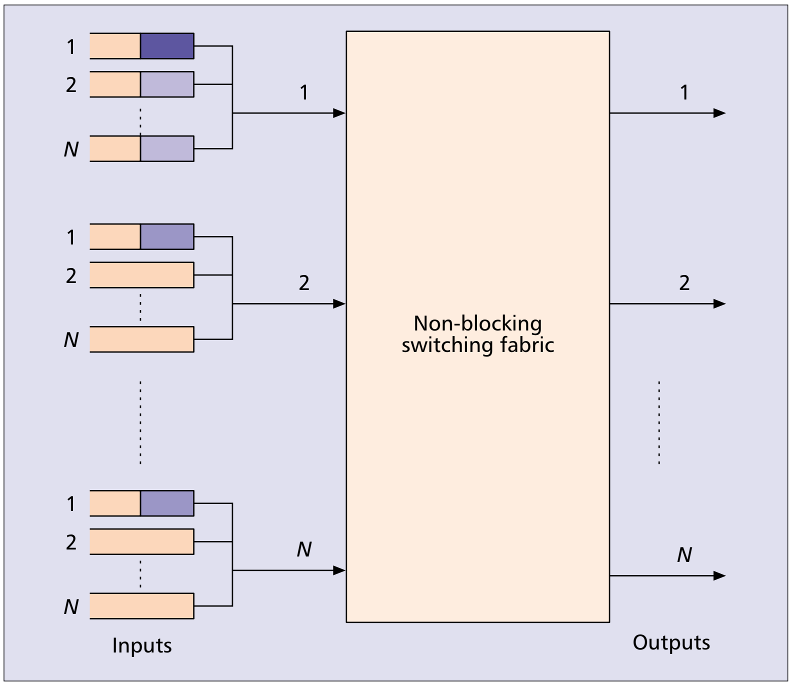
1
2
N
IPC
S_i
IN
S_o
OPC
1
2
N
IPC/OPC: Input/output port controller
IN: Interconnecting network

Figure 1. *A general model for an* N x N *ATM switch.*

Virtual output queuing: This queuing scheme overcomes the HOL blocking associated with FIFO input queuing while keeping its scalability advantage [3]. In this technique each input port maintains a separate queue for each output port. One key factor in achieving high performance using VOQ switches is the scheduling algorithm, which is responsible for the selection of packets to be transmitted in each time unit from the input ports to the output ports. Several algorithms, such as parallel iterative matching (PIM) [3], *i*SLIP [4], and RPA have been proposed in the literature. It was shown that with as few as four iterations of the above iterative scheduling algorithms, the throughput of the switch exceeds 99 percent [5]. As a result, this switch architecture is receiving a lot of attention from the research community, and many commercial and experimental switches based on this queuing technique have already been built such as the DEC Systems AN2 switch [3] and the Tiny-Tera switch [4].

Combined input-output queuing: This queuing scheme is a combination of input and output queuing [2]. It is a good compromise between the performance and scalability of both OQ and input-queued switches. For input-queued switches, at most one packet can be delivered to an output port in one unit of time. For an output-queued switch, up to N packets can be delivered to an output port in one unit of time. Using CIOQ, instead of choosing these two extreme choices, we can choose a reasonable value in between. This can be accomplished by having buffers at both the input and output ports.

Each of these queuing strategies is characterized by certain advantages and drawbacks. The simplest and most scalable approach, however, is input queuing. In this architecture, each input port maintains a FIFO queue of cells, and only the first cell in the queue is eligible for transmission during a given time slot. The drawback of FIFO queuing is that when the cell at the head of



■ **Figure 2.** *Architecture of an* N x N *multiple input queues ATM switch.*

the queue is blocked, all cells behind it in the queue are prevented from being transmitted, even when the output port they need is idle. This is HOL blocking. It was shown through mathematical analysis and computer simulation that HOL blocking limits the throughput of each input port to a maximum of 58.6 percent under uniform random traffic, and much lower than that for bursty traffic [2]. In particular, it has been shown that for exponential packet lengths and Poisson arrivals, the saturation throughput is only 0.5.

Various approaches have been proposed to overcome the problems associated with FIFO input queuing: adopting a switch expansion, a windowing technique, or a channel grouping technique. For more details about these techniques the reader is referred to [2].

In contrast to an input queuing switch, an OQ switch is usually referred to as the ideal switch in terms of its performance on throughput, controlled packet delay, and so on. Input queuing and OQ, however, each has a performance bottleneck the other does not. Recall that an *N* x *N* OQ switch must run the switching fabric at a speed greater than the sum of the speeds of the incoming links. While this may be solved by building "super-fast" interconnection networks, that still leaves the problem of storing packets rapidly in output buffers. The rate at which an output buffer can be accessed is limited by dynamic random access memory (DRAM) or static RAM (SRAM) bandwidths. These ultimately limit the speed at which an OQ switch can run. One way to get around this problem is to place all queuing at the input ports, resulting in an input queuing switch. However, with this approach an arbiter must resolve contention for the switching fabric and input/output ports. It is hard to design arbiters that run at high speeds and can also fairly schedule the switch fabric and output lines. Fortunately, recent research has suggested that this problem may be solvable with current technologies.

In the rest of this article, we will concentrate on the input queuing switch and its related algorithms to schedule the buffered packets. Specifically, VOQ/CIOQ switches are highlighted due to their great potential to achieve performances comparable to OQ switches, without loss of scalability. Before further discusings these algorithms, we first describe the architectures of VOQ/CIOQ switches.

VOQ/CIOQ Switches

In this section we present an overview of the VOQ/CIOQ switch architecture and the scheduling algorithms to find matchings of the input and output ports, which are used to schedule the enqueued packets to be transmitted across the switching fabric. The internal speedup of the switch is assumed to be one unless specified otherwise.

The packet switch under consideration is an *N* x *N* nonblocking switch; that is, the *N* inputs are connected to the *N* outputs via a nonblocking interconnection network (e.g., crossbar switch) [3]. The switch being nonblocking means that a cell may be sent from any input to any output, provided that no more than one cell is sent from the same input and no more than one cell is received by the same output. Each input queue of the switch is a random access buffer. This random access buffer can be used to construct *N* FIFO queues, each of which is used to store the cells destined for one of the *N* output ports. The architecture of this switch is shown in Fig. 2. The first cell in each queue can be selected for transmission across the switch in each time slot, with the following constraints:

- Only one cell from any of the *N* queues in an input port can be transmitted in each time slot.
- Only one cell can be transmitted from the *N* input ports to an output port at any given time slot. In other words, at most one cell could be received by a single output port.

If one buffer per output port is also provided on the output side of a VOQ switch, we have a CIOQ switch. Hence, the results established for the VOQ switches are also applicable to the CIOQ switches unless specified otherwise.

The switch scheduling algorithm that decides, for each time slot, which inputs transmit their queued cells to which outputs is of paramount importance. One such effective algorithm is *parallel iterative matching* [3]. The PIM algorithm cannot support traffic with deterministic QoS guarantees. However, its parallel and iterative running fashion serves as a common basis for a large number of related algorithms designed for VOQ switches (e.g., [4]). For this reason, we first introduce the PIM algorithm and the difficulties in designing algorithms used for VOQ switches to provide QoS guarantees, before further discussing QoS-capable algorithms.

Parallel Iterative Matching and Difficulties

PIM

For switches used in high-performance networks, the switch scheduling algorithm must be able to provide high throughput, low latency, and graceful degradation under heavy traffic loads. Anderson *et al.* [3] considered the architecture of a nonblocking switch with random access buffers (as shown in Fig. 2), and cast the switch scheduling problem as a bipartite matching problem of finding conflict-free pairing of inputs to outputs. The high throughput and low latency of the switch dictates that the scheduling algorithm must be able to find a matching of as many conflict-free pairings as possible using as little time as possible. Note that a *maximum matching* is a matching with the maximum number of paired inputs and outputs; a *maximal matching* is one in which no unmatched input has a queued cell destined for an unmatched output (i.e., no parings can be trivially added). Unfortunately, all the conventional bipartite maximum matching algorithms have high time complexity with regard to the time constraint imposed by high-speed gigabit-per-second links. As a solution, Anderson *et al.* [3] proposed an algorithm, PIM, to find a maximal matching.

The algorithm proposed by Anderson et al. uses parallelism, randomness, and iteration to find a maximal matching between the inputs that have queued cells for transmission and the outputs that have queued packets (at the inputs) destined for them. Maximal matching is used to determine which inputs transmit cells over the nonblocking switch to which outputs in the next time slot. Specifically, their matching algorithm iterates the following three steps until a maximal matching is found or a fixed number of iterations is performed.[1]

Request: Each unmatched input sends a request to *every* output for which it has a queued packet.

Grant: If an unmatched output receives any requests, it grants to one by *randomly* selecting a request uniformly over all requests.

Accept: If an input receives grants, it accepts one by selecting an output among those that granted to this input.

By considering only unmatched inputs and outputs, each iteration only considers connections not made by earlier iterations. In spite of the worst case time complexity of $O(N^2)$, it was shown that under uniform traffic, each iteration will match or eliminate on average 3/4 of the remaining possible connections and thus the algorithm will converge to a maximal match in $O(\log N)$ iterations [3]. Furthermore, both simulation and analysis results showed that a throughput over 99 percent can be achieved by a VOQ switch under uniform traffic and scheduled by the four-iteration PIM algorithm [5], independent of switch size.

A number of *maximum weight matching* algorithms that achieve 100 percent throughput in a VOQ switch under both uniform and nonuniform traffic arrivals were proposed in [6]. In these algorithms, each connection among the inputs and outputs of a switch is assigned a weight. The matching with the maximum weight is found by the algorithms at each time slot to schedule the enqueued packets. Once again, the algorithms can run in an iterative way like PIM. In addition to achieving the highest throughput, some of the proposed algorithms were shown to be starvation-free. However, the main disadvantage of these algorithms is that a time complexity of $O(N^{2.5})$ is required to find a maximum matching in each time slot, which may be too difficult (if impossible) to put into practice under high-speed environments.

Despite the high throughput achieved by these algorithms, they are incapable of supporting traffic with QoS guarantees. In the next section we first introduce the difficulties in designing scheduling algorithms for VOQ switches with predictable performance. Then we present some proposed algorithms and solutions for this problem.

For switches used in high-performance networks, the switch scheduling algorithm must be able to provide high throughput, low latency, and graceful degradation under heavy traffic loads.

DIFFICULTIES

While there has been a lot of research on developing scheduling policies that can provide QoS guarantees and designing scalable high-speed switches, very little has been done to implement these QoS scheduling policies on scalable high-speed switches such as VOQ or CIOQ. Most of the research on providing QoS guarantees on high-speed switches assumes the underlying switch to be OQ or centralized-shared-memory crossbar networks [7]. Given the poor scalability of these switches, these research efforts have very little practical value.

The difference between guaranteeing the QoS in a VOQ/CIOQ switch and doing so in an OQ switch is mainly due to the lower (limited) switching capacity of the switching fabric in a VOQ/CIOQ switch. In an OQ switch, any packet can be available immediately when it is scheduled to be served, because there is no output contention. As a result, guaranteeing the QoS in an OQ switch can be performed by employing suitable service disciplines for each output port independently. The scheduling of packets enqueued in different outputs can be *isolated* from one another. However, the limited switching capacity of a switching fabric in a VOQ/CIOQ switch leads to some packets not being promptly transmitted across the switching fabric when they should receive service. Consequently, the packets which fail to go through the switching fabric lose their chances of being serviced, which may result in violating their QoS.

Therefore, the key point for providing QoS guarantees in a VOQ switch is to design a scheduling algorithm which can guarantee that queued packets are transmitted across the switch fabric promptly. If the delays of queuing packets can be guaranteed, the employed scheduling algorithm will not lead to "starvation" for queued packets at any port.

An intuitive idea as well as the simplest method is to allocate $1/N$ of the bandwidth to each link between a pair of input and output ports, and service each link in a round-robin fashion. However, this allocation is fixed, not scalable, and of reasonable value to uniform traffic only. As a result, algorithms which are flexible and independent of switch size and traffic pattern are sought by various parties from different research directions.

A number of algorithms using different methods to solve this problem have been proposed. In general, the proposed scheduling algorithms

[1] *A three-stage switch similar to this request-grant-accept model is described in sections 6.3 and 6.4 of Hui.*

The WPIM algorithm was proposed for providing bandwidth guarantees in a VOQ switch. WPIM varies from the original PIM algorithm by allowing flexible allocation of bandwidth among different links in a simple manner.

can be classified into three categories according to the matching algorithms used to match inputs and outputs in each time slot: algorithms based on time slot assignment, those based on maximal matching, and those based on stable matching. In the following sections we will explore these algorithms in detail.

Algorithms Based on Time Slot Assignment

To provide guaranteed performance of constant-bit-rate (CBR) traffic in a VOQ switch, a number of bandwidth-reservation-based algorithms have been designed. These algorithms carefully plan the times when the packets have to leave the VOQ switch, which results in the guaranteed-rate property that enables CBR service for real-time traffic with bounded delay.

When the PIM algorithm was proposed in [3], a scheduling algorithm was also proposed for reserving time slots for the provision of bandwidth guarantees for CBR traffic going through a VOQ switch. However, the algorithm is too complicated and limited in its application for CBR traffic only. In addition, only a small portion of the maximum switch bandwidth (throughput) can be reserved by that algorithm (i.e., the bandwidth utilization is too low).

The Weighted Probabilistic Iterative Matching (WPIM) algorithm was proposed in [8] for providing bandwidth guarantees in a VOQ switch. WPIM varies from the original PIM algorithm by allowing flexible allocation of bandwidth among different links in a simple manner. The operations of WPIM consist of two consecutive applications of the original PIM algorithm. Consequently, WPIM has a time complexity of $O(N^2)$ for running once in each time slot. The key to WPIM is computing the weights used for resolving input and output port contentions. Similar to CBR-PIM, bandwidth guarantees in a VOQ switch scheduled by WPIM are achieved by making reservations during the connection setup stage. Simulation results reveal that WPIM is able to achieve probabilistic bandwidth and delay guarantees.

An idling weighted round-robin (WRR) scheduling algorithm has also been proposed for supporting bandwidth guarantees in VOQ switches [9]. The WRR algorithm performs arbitration for queued packets at the connection level by using the Slepian-Duguid method. The time complexity of the WRR algorithm is $O(N^2f)$, where f is the frame size. A maximum throughput of 100 percent can be achieved by a switch scheduled by the WRR algorithm. However, similar to the other connection-level algorithms based on bandwidth reservation, the WRR algorithm is designed with the aim of providing bandwidth for CBR traffic only.

Algorithms performing arbitration at the connection level require a weight to be given for each traffic stream. This given weight is kept unchanged for each traffic stream during the lifetime of that traffic stream. The constant rate of CBR traffic is a natural candidate for the weights. Unfortunately, it is difficult (if not impossible) to explore useful properties from a variable-bit-rate (VBR) traffic stream for computing the weight, which will never change once it is given. As we will see later, algorithms running at the packet level exhibit distinct advantages in supporting VBR traffic with QoS guarantees. However, packet-level algorithms need to be run from time slot to time slot and generally have higher time complexities than their counterparts running at the connection level. A compromise would be designing algorithms to run at the frame level, such as in [10].

The BATCH_TSA algorithm proposed in [10] treats the VOQ switch as a TDMA network, where the switching fabric, inputs and outputs of the VOQ switch correspond to the links, sources, and destinations of the TDMA network, respectively. Consequently, providing bandwidth guarantees in a VOQ switch is translated into the classic time slot assignment (TSA) problem in switching theory, where it has been known that 100 percent efficient algorithms exist for any given TSA problem. Scheduled by a 100 percent efficient algorithm, all packets can be serviced in a frame with size of T slots if the offered load at each input and output is not greater than T packets. Hence, a maximum throughput as high as 100 percent can also be achieved in a VOQ switch using TSA-based algorithms. Furthermore, if traffic shaping is applied to ensure that offered traffic load for any input/output port is not greater than T packets over any T continuous time slots, an incoming packet will be guaranteed to be transmitted across the switching fabric in a delay not greater than $2T$. Shaping traffic in such a way is trivial for traffic to each input port. However, it is a challenging task for shaping traffic to each output port. Another shortcoming of this method is that additional storage space is required to buffer incoming traffic before it is eligible to be scheduled in next timeframe of T time slots.[2]

Some intrinsic drawbacks of frame-based scheduling algorithms are inevitable for algorithms based on TSA (e.g., the bandwidth coupling and rate granularity problems). The former means that a higher bandwidth than the long-term average rate of a bursty delay-sensitive traffic stream must be allocated in order to satisfy the delay requirement. The latter means that a small frame size implies a large minimum allocated rate.

Algorithms Based on Maximal Matching

The high throughput achieved by algorithms based on TSA results in the perfect or maximum matching of inputs and outputs used in the allocation of time slots. However, finding a maximum matching has a time complexity of $O(N^{2.5})$, which may be too complicated to be performed in high-speed environments. Recalling that finding a maximal matching has a relative smaller time complexity of $O(N^2)$, it is naturally suggested to use maximal instead of maximum matching of inputs and outputs. The penalty introduced is that the achievable throughput will be degraded.

[2] *Please note that this frame-based method is actually the so-called stop-and-go scheduling method.*

The Lowest Output Occupancy First Algorithm (LOOFA) proposed in [11] maintains a global input preference list to resolve conflicts occurring at each input port. Moreover, the matching algorithm is a PIM-like algorithm running iteratively. As a result, all advantages of the PIM algorithm are inherited (e.g., high parallelism and low complexity). Therefore, the worst case total time complexity for the LOOFA algorithm to find a maximal matching of inputs and outputs is $O(N^2)$, and the maximum iteration number is N.

The results in [11] show that an internal speedup of 2 is sufficient for a VOQ switch scheduled by these proposed algorithms to be work-conserving. A switch is said to be work-conserving if an output is always in service if there is any queued packet destined to this output. In other words, the provable *deterministic* maximum (normalized) throughput for a VOQ switch scheduled by LOOFA is 50 percent. In addition, bounding the delay for each packet can be carried out on a per-flow basis, and the bounded delay can be expressed as a function of the flow's long-term average rate and burstiness. An internal speedup of 4 or 6 is required to bound packet delays with or without dependence on switch size, respectively. One more elaborated result is that any maximal matching algorithm can ensure asymptotic 100 percent throughput for a VOQ switch with an internal speedup of 4, provided that the traffic burstiness at each input and output is bounded. This reveals that with proper traffic shaping, a minimum throughput of 25 percent can be deterministically guaranteed for all PIM-like algorithms terminated with maximal matching.

It is shown in [10] that a VOQ switch scheduled by the CONTINUOUS_MAXIMAL algorithm (which could be an arbitrary maximal matching algorithm) running in each time slot can bound the delay for each packet by $6(\lambda T - 1/3)$, where λ ″ 1/3, provided that traffic arrivals at each input and output port conform to the linear bounded arrival process of the (λ, T) constraint. With traffic arrivals satisfying the (λ, T) constraint there are at most λT arrivals within any time interval of T time slots. Furthermore, the OLDEST_MAXIMAL algorithm can achieve a smaller delay bound of $2(\lambda T - 1)$, where l ″ 1/2 and the traffic for each input and output port is (λ, T) constrained. The OLDEST_MAXIMAL algorithm finds maximal matching in such a way that any packet not in the matching is blocked by another packet arriving no later than that blocked packet.

The difference between the OLDEST_MAXIMAL matching and CONTINUOUS_MAXIMAL algorithm is that the OLDEST_MAXIMAL algorithm is a prioritized matching algorithm. The OLDEST_MAXIMAL algorithm can be defined as a stable matching algorithm for the bipartite problem by regarding the arrival time of each packet as its preference value (or priority). In the case of the OLDEST_MAXIMAL algorithm, an earlier arrival time means higher priority. The maximum throughput achieved by the OLDEST_MAXIMAL and CONTINUOUS_MAXIMAL algorithms are 1/2 and 1/3, respectively. This suggests that scheduling packets in a prioritized fashion is a potential way to achieve better performance in terms of maximum throughput, maximum bounded delay, and so on. Consequently, it is not surprising that a large number of stable matching algorithms were proposed recently for providing QoS in VOQ/CIOQ switches.

Algorithms Based on Stable Matching

The key point in designing a prioritized algorithm is to define the priorities for packets used to resolve contentions for limited resources. The main task in designing an algorithm based on stable matching of inputs and outputs is therefore to explore useful information from queuing packets and the guaranteed QoS to derive packet priorities. Hence, deriving definitions for queuing packet priorities makes up the core of designing a stable matching algorithm used to schedule queuing packets in VOQ/CIOQ switches.

Studies of using a stable matching algorithm to provide QoS in VOQ/CIOQ switches have been carried out by various researchers [7, 12–14]. The very first result in this area was reported in [12]. It was found that a CIOQ switch with internal speedup 4 and scheduled by the Most Urgent Cell First Algorithm (MUCFA) can exactly emulate an FIFO OQ switch. In a CIOQ switch exactly emulating an OQ switch, the departure process of packets at each output port is the same as that in the emulated OQ switch. Consequently, the problem of providing QoS for traffic in a CIOQ switch can be transformed into guaranteeing the traffic's QoS in the emulated OQ switch. If the traffic's QoS is guaranteed by the emulated OQ switch, the traffic's QoS can also be guaranteed by the CIOQ switch because of the identical departure process of packets in both switches.

Unlike LOOFA (which has only a global input preference list), for a stable matching algorithm an input/output preference list is defined for each input/output port to resolve contentions occurring at that input/output. Instead of using PIM-like algorithms [3] to find matching inputs and outputs, the Gale-Shapley algorithm (GSA), which was first introduced by Gale and Shapley to solve the stable marriage problem, was employed to implement the MUCFA.

The GSA used in the CIOQ switch finds *stable* matching of inputs and outputs based on the defined input and output preference lists. Each input/output preference list ranks the output/input packets in order of preferences. A matching is said to be *unstable* if there is at least a pair of input and output which are not matched, but each prefers the other to its currently matched mate. A matching that is not unstable is called *stable*. The stable matching of inputs and outputs has an important property [7] which forms the basis of designing schemes using stable matching for providing QoS guarantees in a VOQ/CIOQ switch:

> *A packet* $\mathrm{P_{ij}}$ *in a VOQ switch will be scheduled by a stable matching algorithm to be transmitted across the switching fabric* if and only if *no packet ahead of* $\mathrm{P_{ij}}$ *in the preference lists of input* i *and output* j *is switched*,

where P_{ij} denotes a packet destined for the j output port from the ith input port.

LOOFA maintains a global input preference list to resolve conflicts occurring at each input port. Moreover, the matching algorithm is a PIM-like algorithm running iteratively. As a result, all advantages of the PIM algorithm are inherited (e.g., high parallelism and low complexity).

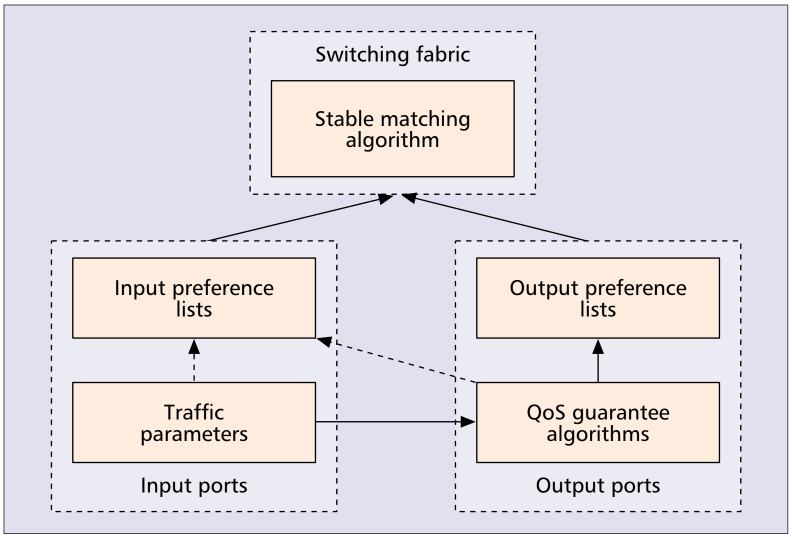


Figure 3. *The framework for providing QoS guarantees in a VOQ switch.*

MUCFA, however, is actually an algorithm to define the input and the output preference lists used by the GSA to find stable matching of inputs and outputs. The input and output preferences of each packet are defined as the number of queuing packets ahead of that packet's clone at the corresponding output buffer of the emulated OQ switch. However, compared to what can be conjectured from the OLDEST_MAXIMAL algorithm, the speedup of 4 required by the CIOQ switch using the MUCFA may be further lowered.

An exciting result reported almost simultaneously in [7, 15][3] is that an OQ switch scheduled by *monotonous* work-conserving service disciplines can be *exactly emulated* by a CIOQ switch with an internal speedup of 2, for any switch size and any *unicast* traffic. A service is said to be *monotonous* if and only if any new arrival doesn't change the relative scheduling order of the packets already enqueued [15]. This monotonous assumption for service discipline employed by the emulated OQ switch is essential for the correctness of results presented in [7, 15], and later in [14], which gives an extensive study on the capacity of stable matching for supporting QoS in VOQ/CIOQ switches.

Different from MUCFA, packets in each input preference list are sorted in reverse order of their arrivals [7, 15], or in increasing order of each packet's output occupancy [7, 15]. A packet's output occupancy is the number of packets waiting in that packet's corresponding output buffer (in the CIOQ switch). In addition, packets in each output preference list are sorted in the same order as in the emulated OQ switch.

While these efforts [7, 12, 15] are a step in the right direction, they have a few shortcomings:

- The methods used to provide QoS guarantees for these switches are developed in the context of unicast traffic only and will no longer be valid after the addition of multicast traffic.
- They achieve QoS guarantees through "exact emulation" of OQ switches [7] which is a conservative methodology under most realistic traffic patterns.
- The algorithms proposed to provide QoS for these switches are too complex to be implemented in real time, especially at high speeds.
- Scheduling variable-length packets was not considered by the proposed algorithms. It was assumed that scheduling would simply be performed on fixed-length packets.

These shortcomings were overcome in [10] based on a systematic and theoretic analysis presented therein. In more detail, the requirements for a CIOQ/VOQ switch to exactly emulate an OQ switch with both unicast and multicast traffic are described in simpler and more intuitive terms. Furthermore, instead of using the GSA which has a time complexity of $\Omega(N^2)$ to find a stable matching of inputs of outputs, the Common Preference Values Stable Matching (CPVSM) algorithm proposed in [14] — which can be used directly to implement MUCFA — indicates a potential way to decrease the time complexity of finding stable matching of inputs and outputs in the context of VOQ/CIOQ switches. To find stable matching of inputs and outputs, the CPVSM algorithm has a complexity equivalent to that of finding a maximal matching: $O(N^2)$. In addition, an algorithm based on stable matching was also proposed to schedule variable-length packets with QoS guarantees.

In summary, a generalized framework for providing QoS guarantees in a VOQ switch using stable matching of inputs and outputs is given in Fig. 3. As shown in the figure, the QoS guarantee in a VOQ switch is specified in terms of OQ. In other words, the referred OQ switch of a VOQ switch is simulated in parallel in order to obtain data for constructing the output preference lists and derive the number of blocking packets in the corresponding input and output preference lists for any packet. The arrows indicate the dependencies among the traffic parameters, the input and output preference lists, the stable matching algorithm, and the QoS guarantee algorithms. The gray boxes show how these elements relate to the hardware parts of the switch. The dashed arrows mean that the input preference lists may be defined over either the traffic parameters or the data provided by the QoS guarantee algorithms. In the former case, the stable matching algorithm is actually traditional GSA. However, when the input and output preference lists are defined using the common data provided by the QoS guarantee algorithms, special properties of input and output preference lists can be explored to design stable matching algorithms with a complexity of $O(N^2)$ (e.g., the CPVSM algorithm) [14]. The reader is referred to [14] for details on how stable matching algorithms with desired properties can be designed according to this generalized framework.

Conclusion

A survey has been carried out on recent trends in designing scheduling algorithms for supporting QoS in input queued switches. Research efforts

[3] *The proof for this result is incorrect in [15]. For more details and to see how the paper is corrected, please go to http://www.cs.cmu.edu/~stoica/IWQoS98-fix.html.*

Algorithms	Complexity	Maximum throughput	Differentiated QoS	Best supported traffic
Time slot assignment	$O(N^{2.5})$	100%	Not supported	CBR
Maximal matching	$O(N^2)$	50%	Not supported	CBR
Stable matching	$\Omega(N^2)$ or $O(N^2)$	50%	Supported	CBR, VBR

Table 1. *Performance comparisons of three classes of algorithms.*

have achieved encouraging results on guaranteeing QoS, including bandwidth, delay, jitter, and so on. Basically, these proposed scheduling algorithms fall into three categories: those based on time slot assignment, those based on maximal matching, and those based on stable matching. The performance of these algorithms in terms of time complexity, maximum achievable throughput, and capability of supporting traffic with differential QoS is compared in Table 1.

With the speed of an input/output port normalized by the internal speedup S, the highest (normalized) throughput as high as 100 percent can be achieved by algorithms based on time slot assignment using maximum matching. However, the proposed algorithms schedule queued packets in a unique fashion and are incapable of providing differential QoS to individual traffic streams. Solving this problem is a good objective for designing new algorithms in this direction.

Instead of performing scheduling at connection-level like algorithms based on time slot assignment do, scheduling algorithms can be run at packet-level which have better performance in environments where traffic is bursty or changing frequently. This leads to designing algorithms based on maximal and stable matchings which have time complexities substantially smaller than the ones based on maximum matching, i.e., $O(N^2)$ vs. $O(N^{2.5})$. Existing results show that a small speedup (e.g., 2–6) and may be with proper traffic shaping, are sufficient for a VOQ switch scheduled by an algorithm either based on maximal and stable matching algorithms to achieve predictable performances in terms of bandwidth, delay, and jitter, and so on.

Among these algorithms, stable matching algorithms present the most attractive attributes in terms of its capability of supporting scheduling algorithms developed for OQ switches to be used in VOQ switches directly. These attributes are highly desired from economical and engineering viewpoints, because scheduling algorithms designed for legacy OQ switches can be directly applied onto VOQ switches without any changes. One benefit is that traffic with differential QoS can be supported in the VOQ/CIOQ switches scheduled by stable matching algorithms.

A lesson learned from these algorithms is that designing scheduling algorithms based on prioritized matching is a feasible way to hit the target of providing QoS for traffic in VOQ/CIOQ switches. Defining scheduling priorities of queuing packets is the key for success in designing an efficient and high-performance algorithm in such a direction.

References

[1] S. Keshav and R. Sharma, "Issues and Trends in Router Design," *IEEE Commun. Mag.*, vol. 36, no. 5, May 1998, pp. 144–51.

[2] R.Y. Awdeh and H.T. Mouftah, "Survey of ATM Switch Architectures," Comp. Networks and ISDN Sys., vol. 27, 1995, pp. 1567–1613.

[3] T.E. Anderson *et al.*, "High-speed switch scheduling for local-area networks," *ACM Trans. Comp. Sys.*, vol. 11, no. 4, Nov 1993, pp. 319–52.

[4] N. McKeown and T. E. Anderson, "A Quantitative Comparison of Iterative Scheduling Algorithms for Input-Queued Switches," *Comp. Networks and ISDN Sys.*, vol. 30, 1998, pp. 2309–26.

[5] G. Nong, J. K. Muppala, and M. Hamdi, "Analysis of Nonblocking ATM switches with Multiple Input Queues," *IEEE/ACM Trans. Net.*, vol. 7, no. 1, Feb. 1999, pp. 60–74.

[6] N. McKeown, V. Anantharam, and J. Walrand, "Achieving 100% Throughput in an Input Queued Switch," *Proc. IEEE INFOCOM '96*, vol. 1, 1996, pp. 296–302.

[7] S. T. Chuang *et al.*, "Matching Output Queuing with a Combined Input Output Queued Switch," tech. rep. Stanford CSL-TR98-758, Mar. 1998.

[8] D. Stiliadis and A. Varma, "Providing Bandwidth Guarantees in an Input-buffered Crossbar Switch," *Proc. INFOCOM '95*, vol. 3, 1995, pp. 960–68.

[9] A. Hung, G. Kesidis, and N. McKeown, "ATM Input-Buffered Switches with Guaranteed-rate Property," *Proc. IEEE ISCC '98*, Athens, Greece, July 1998, pp. 331–35.

[10] T. Weller and B. Hajek, "Scheduling Nonuniform Traffic in a Packet-Switching System with Small Propagation Delay," *IEEE/ACM Trans.Net.*, vol. 5, no. 6, Dec. 1997, pp. 813–23.

[11] P. Krishna *et al.*, "On the Speedup Required for Work-Conserving Crossbar Switches," *IEEE JSAC*, vol. 17, June. 1999, pp. 1057–66.

[12] B. Prabhakar and N. McKeown, "On the Speedup Required for Combined Input and Output Queued Switching," tech. rep. Stanford CSL-TR-97-738, Nov. 1997.

[13] A. Kam and K.-Y. Siu, "Linear Complexity Algorithms for QoS Support in Input-queued Switches with No Speedup," *IEEE JSAC*, vol. 17, June 1999, pp. 1040–56.

[14] G. Nong and M. Hamdi, "Providing QoS Guarantees for Both Unicast and Multicast Traffic in High-Speed Packet Switches with Multiple Input Queues," *GLOBECOM '99*, 1999.

[15] I. Stoica and H. Zhang, "Exact Emulation of an Output Queuing Switch by a Combined Input and Output Queuing Switch," *IWQoS '98*, 1998.

Biography

Mounir Hamdi [M] (hamdi@cs.ust.hk) received his B.S. degree in computer engineering (with distinction) from the University of Southwestern Louisiana in 1985, and M.S. and Ph.D. degrees in electrical engineering from the University of Pittsburgh in 1987 and 1991, respectively. He has been a faculty member in the Department of Computer Science at the Hong Kong University of Science and Technology since 1991 where he is now associate professor of computer science and director of the Computer Engineering Program. In 1999 and 2000 he held visiting professor positions at Stanford University and the Swiss Federal Institute of Technology. His general areas of research are networking and parallel computing. He has graduated more than 10 M.S./Ph.D. students in this area of study. Currently, he is working on high-speed networks including the design, analysis, scheduling, and management of high-speed switches/routers, and WDM networks/switches. He is a member of ACM.

Research efforts have achieved encouraging results on guaranteeing QoS including bandwidth, delay, jitter, etc. These proposed scheduling algorithms fall into three categories: algorithms based on time slot assignment, algorithms based on maximal matching and algorithms based on stable matching.