Providing QoS Guarantees for Unicast/Multicast Traffic with Fixed/Variable-Length Packets in Multiple Input-Queued Switches *

Ge Nong
STMicroelectronics Ltd.
16/F, Tower 1, The Gateway
25 Canton Road, Kowloon, Hong Kong
ge.nong@st.com

Mounir Hamdi
Department of Computer Science
The Hong Kong Univ. of Science and Technology
Clear Water Bay, Kowloon, Hong Kong
hamdi@cs.ust.hk

Abstract

With a deep understanding on the properties of stable matching in the context of multiple input-queued switches, we propose the efficient schemes to guarantee the QoS of any unicast and multicast traffic with fixed- or variable-length packets in an unified way. Using these schemes, the QoS of fixed- or variable-length unicast and multicast packets can be guaranteed by independently *employing suitable service disciplines at the packets' destined outputs like what is being done in an output queueing switch. One of our proposed schemes uses totally $4N$ input/output buffers, each with an internal speed-up of 2 independent of N, for an $N \times N$ switch to support any fixed- or variable-length multicast and unicast packets with OoS guarantees.*

1 Introduction

While there has been a lot of research on developing scheduling policies that can provide QoS guarantees and on 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 output-queued or centralized-shared-memory crossbar networks [8]. Given the poor scalability of these switches, these research efforts have very little practical value.

Recently, some efforts have been reported on how to provide QoS guarantees on VOQ or CIOQ switches [2, 3, 5, 7] [1]. All of the scheduling algorithms developed in these proposals are based on the "stable matching" concept because of its potential in bounding a packet's maximum switching delay.

In this paper, we expand upon these previous research efforts and we address some of their shortcomings. The remainder of this paper is organized as follows: the architectures of VOQ and CIOQ switches are described in section 2. In Section 3, a brief review on some representative of the existing algorithms for guaranteeing QoS in VOQ or CIOQ switches is given. In section 4, our schemes using stable matching for providing QoS guarantees to unicast and multicast traffic with fixed or variable length packets in CIOQ switches are presented. Finally, a conclusion is given in Section 5.

2 Switch Architecture

The switches under investigations in this paper are assumed to be non-blocking $N \times N$ switches. The architecture of a VOQ switch is shown in Fig. 1. The switch operates at discrete time slots and the (unicast or multicast) packets arrive only at the beginning of time slots and depart only at the end of time slots. The inputs and outputs of the VOQ switch are connected via a nonblocking interconnection network such as a crossbar. Each input maintains a separate queue for packets destined to each output port. Only the HOL packet of each queue can be selected for transmission across the switch in each time slot, with the following constraints:

1. Each input can transmit at most one HOL packet from any of its N queues, in each time slot.
2. Each output can receive at most one packet in each time slot.

*This work was supported in part by the Hong Kong Research Grant Council under the Grant RGC/HKUST 6026/97E. Most of this work was done when Ge Nong had his Ph.D program in the Hong Kong University of Science and Technology.

[1]There is one problem with one of the algorithms in [7]. Please refer for more details of the algorithm and how it was corrected to the web site http://www.cs.cmu.edu/~stoica/IWQoS98-fix.html.

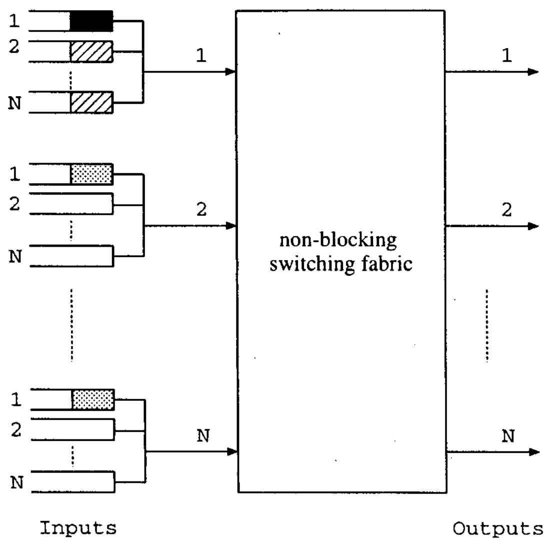

Figure 1. The VOQ switch architecture.

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. When output queueing is needed on the output side, the term CIOQ is used.

Along with an $N \times N$ VOQ switch, we define a referred $N \times N$ OQ switch. Each output in the referred OQ switch is denoted as a *referred OQ server*. The referred OQ switch of a VOQ switch is simulated in parallel with the VOQ switch in order to produce informations needed for guaranteeing the traffic's QoS in the VOQ switch, as will be shown later.

In the rest of this paper, the following notations are used to simplify and clarify our presentation.

- ω: the internal speed-up of the switch normalized by the input/output link speed of 1 cell per time slot. The internal speed-up occurs at the interfaces from each input port to the switch and from the switch to each output port.
- P_{ij}: a packet destined for output j from input i.
- $T(P_{ij})$: the arriving time slot of packet P_{ij}.
- $D(P_{ij})$: the delay of the packet P_{ij} in the referred OQ server. The delay of a packet includes both the waiting time and the service time.
- $D_{VOQ}(P_{ij})$: the delay of the packet P_{ij} in the VOQ switch, or the delay of the packet P_{ij} in the input buffer of the CIOQ switch.
- $B_{in}(P_{ij})$: the maximum number of packets that may block P_{ij} and are destined for not output j from input i.
- $B_{out}(P_{ij})$: the maximum number of packets that may block P_{ij} and are destined for output j from all inputs.
- $Q(i,j)$: the queue for packets destined to the output port j from the input port i.

Before presenting our solutions, we have a brief review on some representatives of the existing algorithms attempting the problem of providing QoS to unicast fixed-length packets in VOQ/CIOQ switches.

3 Existing Algorithms

It was proved in [3] that an OQ switch scheduled by *monotonous* work-conserving service disciplines can be *exactly emulated* by a CIOQ switch with an internal speed-up of 2, for any switch size and any *unicast* traffic. The departure process of packets in a CIOQ switch exactly emulating an OQ switch is the same as that in the emulated OQ switch. A service principle is said to be *monotonous* if and only if any new arrival doesn't change the relative scheduling order of the packets already enqueued [7]. In this paper, we also assume that the service disciplines are monotonous. This assumption is essential for the correctness of results presented in this paper as well as the other papers [3, 7]. Each output of the emulated OQ switch independently schedules the service of the offered traffic by the employed monotonous work-conserving service discipline. Consequently, QoS guarantees can be provided in the CIOQ switch by emulating an OQ switch.

Instead of using PIM-like algorithms [1] to find matching of the inputs and outputs, the Gale-Shapley algorithm (GSA), which was first introduced by Gale and Shapley to solve the stable marriage problem, was employed by the CIOQ switch [3]. 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 outputs/inputs packets in order of preferences. A matching is said to be *unstable* if there is at least a pair of input and output which aren't matched, but which each prefer the other to their currently matched mates. A matching which is not unstable is called *stable*.

The stable matching of inputs and outputs holds an important property [3] which forms the basis of designing schemes for providing QoS guarantees in a VOQ/CIOQ switch.

Proposition 1 *A packet 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 P_{ij} in the preference lists of input i and output j is switched.*

Proof: The correctness of Proposition 1 comes from the definition of stable matching.

The results[2] established in [3, 7] were developed in the context of *unicast traffic only* and won't be valid any more in case multicast traffic is included. However, the ability to support multicast traffic is indispensable for high-speed packet switches designed for the future Internet. In the following section, we will address the problem of providing QoS guarantees for both unicast and multicast traffic with fixed and variable-length packets in CIOQ switches using stable matching scheduling.

4 Main Results

Based on the defined input and the output preference lists, a stable matching algorithm is performed to schedule enqueued packets to be transmitted across the switching fabric. Provided that the switching fabric is fast enough to transmit all packets to their destined outputs promptly, the specified QoS of packets are guaranteed. The question is: How fast should the switching fabric be in order to guarantee that all packets can be transmitted to their destined outputs promptly? The answer is given by the required internal speed-up. To proceed further, we first introduce the definition for output preference lists.

Definition 1 *An output preference list is defined according to the order that packets receive service in the referred OQ server.*

With this definition of output preference list, it is obvious that $B_{out}(P_{ij}) = D(P_{ij}) - 1$ when the service discipline employed by output j is monotonous and the CIOQ switch exactly emulates an OQ switch. This is because that with monotonous service discipline, the enqueued packets' relative orders cannot be changed by the new arrivals. As a result, there are at most $D(P_{ij}) - 1$ packets ahead of P_{ij} in output j's preference list. However, when the service discipline is not monotonous, the relative orders of enqueued packets may be changed by the new arrivals. Consequently, the packets ahead of P_{ij} in the output j's preference list cannot be bounded. This is why we assumed that the service disciplines must be *monotonous*.

However, $B_{in}(P_{ij})$ is not immediately obtainable for any packet P_{ij}. Let $B(t)$ be a function of time which accumulates the total number of blocking packets of P_{ij}, including which arriving before $T(P_{ij})$ and which arriving within the time interval $[T(P_{ij}), t]$, $t < T(P_{ij}) + D(P_{ij}) + X$. $B(t)$ should be *non-decreasing* when t increases. If the switch is fast enough to transmit these $B(t)$ packets across the switching fabric within the time interval $[T(P_{ij}), t]$, the QoS of P_{ij} can be guaranteed. The relationship among $B(t)$, $D(P_{ij})$ and ω is illustrated in Fig. 2. In this figure, $B(0) - B_{out}(P_{ij})$ is the maximum number of packets arriving at input i earlier than P_{ij} and may block P_{ij}. The theorem below is a result indicating such a relationship.

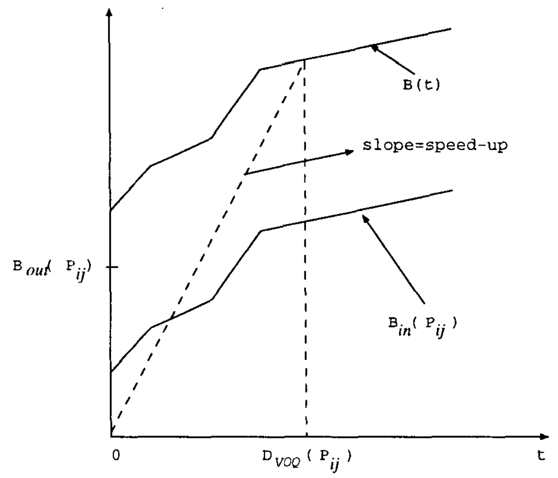

Figure 2. The relationship among traffic arrivals, packet delay and required internal speed-up.

Theorem 1 *In a VOQ/CIOQ switch with the output preference lists as defined by Definition 1, if the service discipline employed by output j is* monotonous *and the speed-up* $\omega \geq 1 + \frac{B_{in}(P_{ij})}{D(P_{ij})+X}$ *for any packet* P_{ij}, $\forall i \in [1, N]$, *then any packet* P_{ij} *can be transmitted across the switching fabric within a delay of* $D(P_{ij}) + X$.

Proof: To bound the delay of any packet P_{ij} by $D(P_{ij}) + X$, we must have $B_{out}(P_{ij}) = D(P_{ij}) + X - 1$. As a result, the number of packets ahead of P_{ij} in the preference lists of input i and output j is now upper bounded by $D(P_{ij}) + X - 1 + B_{in}(P_{ij})$. For a maximum delay of $D(P_{ij}) + X$, the required speed-up is

$$\omega \geq \frac{D(P_{ij}) + X + B_{in}(P_{ij})}{D(P_{ij}) + X} = 1 + \frac{B_{in}(P_{ij})}{D(P_{ij}) + X} \quad (1)$$

In Theorem 1, the definition of $B_{in}(P_{ij})$ was not explicitly specified, which leaves us room for defining the input preference lists in various ways. The various definitions of input preference list may lead to different values of $B_{in}(P_{ij})$ – which may result in different time complexities for maintaining input preference lists and finding the stable matching of inputs and outputs. We next use a definition for the input preference lists (i.e., LIFO) proposed in [3] as a

[2] A CIOQ switch with an internal speed-up of 2 can exactly emulate an OQ switch by using stable matching.

sample for the application of Theorem 1.

An Example: LIFO Input Preference Lists
In [3], the problem of exactly emulating an OQ switch by a CIOQ switch was investigated, where the input preference lists are defined as follows:

Definition 2 *The packets in an input preference list are sorted in the reverse order of packets' arrivals, i.e., LIFO.*

With Definition 2, $B_{in}(P_{ij})$ is equal to the queuing delay of P_{ij} in the input buffer. For a CIOQ switch exactly emulating the referred OQ switch, this delay is upper bound by $D(P_{ij}) - 1$. Hence, we have $B_{in}(P_{ij}) = B_{out}(P_{ij}) = D(P_{ij}) - 1$ for any packet P_{ij}. Using Eq (1), we have $\omega \geq 2 - \frac{1}{D(P_{ij})}$ which produces $\omega \geq 2$ when $D(P_{ij}) \to \infty$. This required speed-up 2 for a CIOQ switch to exactly emulate the referred OQ switch has already been derived by [3] in another different way. We prove it here just for the demonstration purpose of the usage of Theorem 1.

4.1 Reducing The Required Internal Speed-up

With Definition 2 for the input preference lists, $B_{in}(P_{ij})$ is upper bounded by $D_{VOQ}(P_{ij}) - 1$ for any packet P_{ij}. This implies that the required speed-up of the buffers can be further decreased by splitting each input buffers into multiple ones so that the rate of packet arrivals to each individual buffer is lowered. Decreasing the required speed-up for the buffers is extremely important. For example, the research on the implementation of the Tiny-Tera has pointed out that the bottleneck of the Tiny-Tera's speed is the memory (used in buffers) bandwidth and the scheduler speed [6]. Hence, reducing the required memory bandwidth is highly desirable.

Fig. 3 shows a sample architecture in which the required internal speed-up is 1.5. As we notice from the figure, two (independently) parallel switching fabrics are provided. The buffer at each input port is halved into two (equal size) sub-buffers and each sub-buffer can be accessed (read/written) independently. It is assumed that the maximum arrival rate at each input port is 1 cell per time slot. The demultiplexer at each input port dispatches the incoming cells to its two attached sub-buffers alternately so that the maximum arrival rate at each sub-buffer is 0.5 cell per time slot. It is not difficult to show that under the given conditions, $B_{in}(P_{ij}) = 0.5D(P_{ij}) - 1$ for any enqueued packet P_{ij}. Recalling the results stated in Theorem 1, the required speed-up for each sub-buffer, switching fabric and output buffer is 1.5. Or in other words, the aggregate internal speed-ups for each input port and output port are 3 and 1.5, respectively.

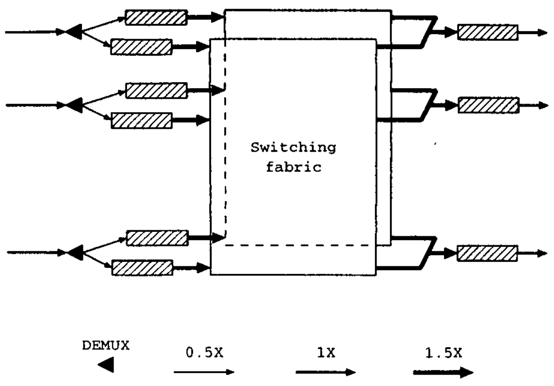

Figure 3. A sample architecture with the required internal speed-up of 1.5.

4.2 Support of Multicast Packets

As can be seen from our derivation of $B_{in}(P_{ij})$ and $B_{out}(P_{ij})$, the assumption of *unicast traffic only* is essential to the correctness that a speed-up of 2 suffices for a CIOQ switch to exactly emulate an OQ switch. With unicast traffic only, at most one (unicast) packet can arrive at any input at each time slot so that there are at most $D(P_{ij}) - 1$ packets enqueued in input i that can block P_{ij}. In case that multicast traffic is included, bounding $B_{in}(P_{ij})$ for any packet P_{ij} get more complicated as analyzed below.

Our solution for scheduling multicast packets is stimulated by the following observation. A CIOQ switch with an internal speed-up of 2 can exactly emulate an OQ switch. As a result, there are at most 2 packets are transmitted to an output port at each time slot. If a transmitted packet is a multicast packet and has remaining instance(s) to other destined output ports, we can re-circular this packet to an input buffer to participate in subsequent scheduling. As the total number of packets transmitted to their output ports at each time slot is at most $2N$, we can introduce $2N$ additional input buffers to bound that there is at most one re-circular packet to each at a time slot. Hence, we use 3 buffers at each input port, each with an internal speed-up 2. The switch architecture enabling our multicast scheduling scheme is shown by Fig. 4. The DEMUX at each output port distributes re-circular multicast packets to each connected input buffer evenly so that no input buffer will receive more than one re-circular packet at each time slot. The switching fabric is now a $3N \times N$ running at an internal speed-up of 2. As noticed from the figure, each input/output buffer operates in an internal speed-up of 2. Therefore, each input and output port has an aggregate internal speed-up of 6 and 2, respectively.

The detailed unified scheduling algo-

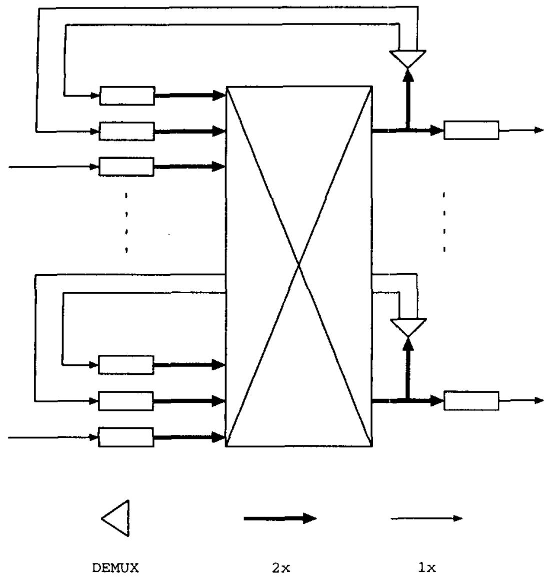

Figure 4. The architecture of our proposed multicast switching.

rithm for both unicast and multicast packets is given below.

Unified Scheduling Algorithm

Step 1: Perform the original unicast stable matching algorithm over all packets queuing at all input buffers to select a set of packets for transmissions to their destined output ports;

Step 2: Transmit the selected packets to their destined output ports. If multiple instances of a multicast packet are selected simultaneously to be forwarded to their destined output ports, only one remains as a multicast packet with the forwarded instances being deleted from the fan-out pattern, all the others are unicast copies;

Step 3: If a packet transmitted to an output port is a multicast packet and has remaining instance(s) to other output port(s), re-circulate it to an input buffer via the DEMUX at this output port.

Scheduled by the above algorithm, no more than one packet can arrive at each input buffer at a time slot. Consequently, it is immediately seen that the scheduled switch with an internal speed-up of 2 can exactly emulate an OQ switch. Comparing to an unicast CIOQ switch, re-circulating unfinished multicast packets back to the input buffers can lead to two new problems, i.e., re-ordering of input/output preference lists and re-sequencing of cells arriving at the switch from the same flow. Addition efforts are required to solve these two problems for practical use of the proposed multicast scheme, that will be addressed elsewhere due to the limited space.

4.3 Scheduling Variable-Length Packets

So far, we have been concentrating on using stable matching to schedule fixed-length packets in a VOQ/CIOQ switch. However, there is a recent trend in building high-speed IP routers over ATM switches for the next generation internet, i.e., the so-called ATM under IP or IP switching. When a VOQ switch is used as the underlying core switch of an IP router, the arriving TCP/IP packets are naturally of variable-length. Hence, segmenting IP packets into cells and re-assembling the cells into IP packets are needed before and after packets' transmissions across the switching fabric because cells belonging to various packets may interleave with one another. Such a process can be time consuming and may not be easily implementable at very high speed. One way to eliminate this fragmentation and re-assembly is to use variable-length packet scheduling instead of fixed-length cell scheduling, so that each HOL packet can be transmitted across the switching fabric in a whole without being interrupted (interleaved) – consecutive cells belong to the same IP packet – by others using the same transmission path. That is, the scheduling of variable-length packets is *non-preemptive and non-interleaving*.

In more details, a packet P_{ij} will be transmitted across the switching fabric without being interrupted by any other packets destined to the output j from the input i. Therefore, at most N reassembly queues are needed to be maintained in an output port—each queue separately buffer packets routed from one of the N inputs. As a result, a packet will be stored at its corresponding output port buffer in continuous positions. Which in turn releases the switch from the need of re-assembling cells into packets at each output port. That is especially important for a CIOQ switch with an internal speed-up greater than one.

Lack of the non-interleaving transmissions of packets, packets on the same flow/circuit may be transmitted to the destined output port in an order different from their arrivals at the input port. Therefore, a re-assembly buffer must be maintained for each flow/circuit. In addition, each arriving cells of a packet at an output port must find its location in the re-assembly buffer, that generally requires a sorting operation whose complexity is difficult to be bounded (the sorting complexity is naturally depended on the number of buffered packets at the re-assembly buffer, that is generally difficult to be bounded.)

Without loss of generality, it is assumed that a variable-length packet consists of one or multiple (integral) fixed-length packets, or so-called *cells*. If we directly apply a cell based stable matching algorithm to schedule enqueueing variable-length packets in a VOQ/CIOQ switch, a newly arriving packet may interrupt the transmission of another packet. As a result, interleaving of packets' transmissions will occur. To avoid this problem, the scheduling algorithms can be modified accordingly as follows.

- Once a packet P_{ij} is scheduled to be transmitted across the switching fabric, it is set to be the Virtual HOL (VHOL) packet of $Q(i,j)$.
- Whenever a HOL packet of $Q(i,j)$ is scheduled by the employed stable matching algorithm to be transmitted across the switching fabric, the VHOL packet of $Q(i,j)$ is transmitted instead of the HOL packet of $Q(i,j)$.

Let L_{max} denote the maximum packet length measured in number of cells, we have the following lemma for scheduling variable-length unicast packets in a VOQ/CIOQ switch.

Lemma 1 *For a VOQ/CIOQ switch with an internal speed-up $\omega \geq 3$ under unicast traffic with variable-length packets, if the inputs and the outputs preference lists are defined by Definition 2 and Definition 1, then $D_{VOQ}(P_{ij}) \leq D(P_{ij}) + L_{max}$.*

Proof: To have $D_{VOQ}(P_{ij}) \leq D(P_{ij}) + L_{max}$, Fig. 5 shows how to develop the required speed-up. It is actually to solve the below equations.

$$\begin{aligned} B(t) &= D(P_{ij}) + 2L_{max} + t \\ Y(t) &= \omega t \\ B(t) &= Y(t) \\ t &\leq D(P_{ij}) + L_{max} \\ D(P_{ij}) &\geq 1 \end{aligned}$$

Algebra operations can result in $\omega \geq 3$.

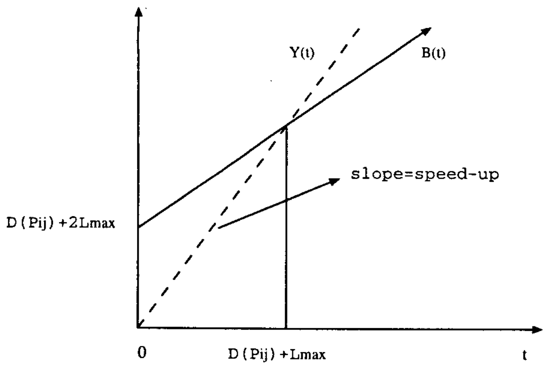

Figure 5. The required speed-up for variable-length packet scheduling.

5 Conclusion

We have concentrated on the ability of stable matching to support any multicast and unicast traffic with *deterministic* guaranteed QoS in VOQ/CIOQ switches. Un-interleaving transmissions of variable-length multicast packets remains an open problem. In addition, given the input and the output preference lists at each time slot, the complexity of stable matching, that has the time complexity of $\Omega(N^2)$ [4], currently constitutes the bottleneck of the switch scalability.

References

[1] T. E. Anderson, S. S. Owicki, J. B. Saxe, and A. P. Thacker. High-speed switch scheduling for local-area networks. *ACM Transactions on Computer Systems*, 11(4):319–352, Nov 1993.

[2] A. Charny, P. Krishna, N. Patel, and R. Simcoe. Algorithms for providing bandwidth and delay guarantees in input-buffered crossbars with speed up. In *IWQOS '98*, 1998.

[3] S. T. Chuang, A. Goel, N. McKeown, and B. Prabhakar. Matching output queueing with a combined input output queued switch. Technical report, Stanford CSL-TR-98-758, Mar. 1998.

[4] D. Gusfield and R. W. Irving. *The stable marriage problem.* The MIT Press, 1989.

[5] P. Krishna, N. Patel, A. Charny, and R. Simcoe. On the speedup required for work-conserving crossbar switches. In *IWQOS '98*, 1998.

[6] N. McKeown, M. Izzard, A. Mekkittikul, W. Ellersick, and M. Horowitz. Tiny tera: a packet switch core. *IEEE Micro*, 17(1):26–33, Jan.-Feb. 1997.

[7] I. Stoica and H. Zhang. Exact emulation of an output queueing switch by a combined input and output queueing switch. In *IWQoS'98*, 1998.

[8] H. Zhang. Service disciplines for guaranteed performance service in packet-switching networks. *Proceedings of the IEEE*, 83(10):1374–96, Oct. 1995.