

The Limits of Input-Queued Switch Performance with Future Packet Arrival Information

Timothy X Brown*
Electrical & Computer Engineering
timxb@colorado.edu

Harold N. Gabow
Computer Science
hal@cs.colorado.edu

University of Colorado
Boulder, CO 80309-0425

Abstract

This paper investigates the benefit of a hypothetical oracle that gives future packet arrival information in a fixed-size-packet switch. While future information has no effect on output-queued switch performance, this paper shows in simulation, the input-queued switch with no switch fabric speed up has wait-time and buffer-size performance close to that of the output-queued switch even with only a few time slots of future information. This finding suggests that the performance difference between input and output-queued switches may be smaller than previous studies suggest. This paper investigates the theoretical aspects of using future information and shows the following. Input-queued switches are proved equivalent (in number of packets sent) to output-queued switches for 2 inputs and 2 outputs. A linear-time algorithm is presented for achieving this equivalence. Larger switches are strictly not equivalent, and optimally using future information is in the worst case NP-hard. A heuristic that simplifies using the future information is investigated and found to be sub-optimal in rare instances for larger switches while optimal for 2×2 switches.

Keywords: Switching systems, Packet switching, Input queueing, Output queueing, Complexity theory.

Submitted to *Computer Networks* 5/01

Revised 5/02

Revised 11/02

Accepted 1/03

*Research supported by NSF CAREER Award NCR-9624791.

1 Introduction

High-speed packet switches and routers are built around a non-blocking switch fabric that sends fixed-size arriving packets to their destination ports (e.g. [20]). Since one output port can be the destination for multiple packets, some packets must queue in the switch to be sent later (so-called *output blocking*). Therefore, switch performance depends not only on the switch fabric speed, but on the queueing and arbitration that decides which packets to send and which to buffer.

An output-queued (OQ) switch sends all packets directly to queues at their destination output. OQ has the best mean wait-time and queue-size performance [13]. Buffer sizes (per input) for a given packet loss rate are dramatically reduced when the OQ share a single large buffer [6]. An OQ shared-buffer switch has the best queueing performance and thus is our benchmark switch. Unfortunately, the switch fabric must be many times faster than the input port speed in order to guarantee that all packets reach their destination.

An input-queued (IQ) switch queues blocked packets at their input port. Given a set of queued packets, a non-blocking subset is chosen by a *packet arbitrator*. The packet arbitrator is central to IQ performance and has been researched extensively (see references). Switch fabric speed ups improve performance [5, 17, 18], but this results in combined IQ and OQ, and we wish to focus on pure IQ. Although IQ has no equivalent to shared buffers, we compare the segregated IQ and shared-buffer OQ directly. Others have argued the best IQ performance is with weighted-matching based arbitrators [15, 24]. These schemes are too complex to be implemented at high speeds. Simpler arbitrators that approximate the optimal controller have been proposed [3, 11, 16, 19, 25, 26, 27]. But even with the best controller, the OQ switch has better performance. Therefore, further improving IQ switch performance requires a different approach. This paper explores using future arrival information.

While future information has no effect on OQ switch performance, it may improve the IQ arbitrator's decisions. The problem is two-fold: predicting future arrivals, and using the predictions to improve switch performance. The predictions may come from several sources.

Connection-oriented protocols give future traffic information useful for arbitration [24]. For example, ATM and MPLS signal future traffic flows [21, 28] which can indicate switch loading. Data traffic is characterized by long-range dependencies [9, 14], so that even with connectionless traffic, useful time-series predictions can be made (e.g. [1, 2, 8]). The first packet in a burst can indicate the number of following packets on the same route. A multi-stage switch can signal arrivals at earlier stage buffers to later stage buffers. Some switches route packets blocked by contention back to inputs [10]. In optics, the mismatch between optical speeds and header processing time has suggested models where packet headers are sent well in advance of the actual packets [22], providing explicit future information.

Practical methods for using the information are also possible. Several of the high-speed implementations can prioritize the arbitration. The future information could bias the priorities towards taking packets from queues that will be loaded in the future. This notion can be formalized as a Markov decision process and with this formulation significant improvements in IQ switch performance have been shown [4]. Optical implementations propose using future information to reduce congestion and meet quality of service objectives [23, 29].

While future information is available in various forms and it has been used in some instances, what are the limits to how much future information can help us? This paper focuses on providing a theoretical foundation. The paper answers three basic questions. *Can a perfect predictor improve IQ switch performance?* Section 4 describes experiments with future knowledge which show IQ and OQ wait times are within a few percent. Buffer performance is also similar. Furthermore, having information only a few time steps into the future provides the majority of the gains. *Are IQ and OQ switches equivalent with future information?* Section 5 shows they are strictly not equivalent for 3×3 and larger switches, but, for 2×2 switches, they can always send the same number of packets in each time slot. *What is the computational complexity of optimally using the future information?* Section 6 shows 2×2 switches have a simple algorithm that is $O(1)$ complexity per time slot. Larger switches have a polynomial algorithm with a large exponent or are NP-Hard, depending on the constraints. Section 7 analyzes a heuristic that can speed up the packet arbitration.

First we begin with the problem formulation and then performance criteria.

2 Problem Formulation

This section describes the packet arbitration problem. In an $N \times N$ switch, fixed-size packets arrive in periodic time slots at one of N input ports destined for one of N output ports. The switch queues alternate between sending existing packets and adding arriving packets. The IQ switch uses virtual output queues where each input has N FIFO sub-queues, one for each output [3, 15]. An input sends a packet from at most one of its sub-queues in each time slot. The sub-queues keep the packets in sequence between a given input-output pair. Similar to IQ, OQ can decompose each output queue into N logical sub-queues. Thus, a single notation can represent the IQ or OQ switch state just before the packets are sent. Let $q = \{q_{io}\}$ where q_{io} is the number of queueing packets that arrived at input i destined for output o . The rows of q correspond to the IQ switch buffers, i.e. input i has $q_i^{in} = \sum_{o=1}^N q_{io}$ packets waiting. The columns of q correspond to the OQ switch buffers, i.e. output o has $q_o^{out} = \sum_{i=1}^N q_{io}$ packets waiting.

Packet arrivals are denoted $a = \{a_{io}\}$ where $a_{io} = 1$ if a packet arrives at q_{io} and $a_{io} = 0$ otherwise. At most one packet can arrive at an input in a time slot (the *arrival constraint*):

$$\sum_{o=1}^N a_{io} \leq 1 \quad \forall i. \quad (1)$$

The packets sent are denoted $s = \{s_{io}\}$ where $s_{io} = 1$ if a packet in q_{io} is sent and $s_{io} = 0$ otherwise. The number of packets sent is $|s| = \sum_{i=1}^N \sum_{o=1}^N s_{io}$. Clearly, $s_{io} = 1$ only if $q_{io} > 0$. Both IQ and OQ switches have the *output constraint*:

$$\sum_{i=1}^N s_{io} \leq 1 \quad \forall o \quad (2)$$

(i.e. each switch output can send at most one packet). IQ switches, have the *input constraint*:

$$\sum_{o=1}^N s_{io} \leq 1 \quad \forall i. \quad (3)$$

(i.e. each input queue can send at most one packet through the switch fabric). A permutation matrix, $p = \{p_{io}\}$ where $p_{io} \in \{0, 1\}$, satisfies the input and output constraints with

equalities. In particular, each row and each column has exactly one one.

Every p defines a non-blocking s for an IQ switch:

$$s_{io} = \begin{cases} 1 & \text{if } p_{io} = 1 \text{ AND } q_{io} > 0 \\ 0 & \text{otherwise} \end{cases}, \quad (4)$$

and we say permutation p was sent. It follows that the packet arbitrator has as many as $N!$ non-blocking s to choose from.

Let the current queue state be q . Let an oracle tell us the next f arrivals, $A = \{a^1, \dots, a^f\}$, where a^t are the arrivals t steps in the future. Similarly, let $S = \{s^0, \dots, s^f\}$ be a packet send sequence. A triple (q, A, S) induces a sequence of queue states where $q^0 = q$ and $q^{t+1} = q^t - s^t + a^{t+1}$, $1 \leq t < f$. Since a^{f+1} is unknown and does not affect which packets can be sent from q^f , we define $q^{f+1} = q^f - s^f$. Thus, the next $f + 1$ sends can be analyzed knowing the next f arrivals (just as when $f = 0$ we can evaluate the packets sent in the current time slot not knowing any future arrivals).

3 Performance Criteria

The packet arbitrator requires some criteria for choosing a set of packets, s^0 to send now. This section defines our global performance criteria from which we define an objective $J(q, A, S)$ for ranking the send sequences, S . With this objective the packet arbitrator chooses an S that maximizes J and sends $s^0 \in S$.

What is our global performance criteria? Since the OQ switch has the best performance, we might require the IQ switch to reproduce any sequence of sends the OQ switch is capable of, but this is not generally possible (see Section 5). Arbitrators can address issues such as differential quality of service, fairness, and priorities. These issues, though important, are not always well defined and can bring in higher issues of resource allocation, admission control, and flow control that cloud the theoretical questions addressed here. Instead, we assume all packets are equal and we measure aggregate performance. Two key aggregate performance measures are the mean wait time (and queue size¹); and the required buffer size

¹The mean wait time and queue size only differ by a constant via Little's theorem, $W = \lambda Q$. So, comments on the relative performance of the mean wait-time apply equally to the mean queue size.

for a given packet loss rate (an outlier metric).

Given this global criteria, potential objectives include:

$$\begin{aligned}
J_a(q, A, S) &= \sum_{t=0}^f (f - t + 1) |s^t| && \text{(minimum average statistics)} \\
J_s(q, A, S) &= \sum_{t=0}^f |s^t| && \text{(send most packets)} \\
J_q(q, A, S) &= -\max\{q_{\max}^{in,t} : t = 1, \dots, f + 1\} && \text{(minimum longest input queue)} \\
J_l(q, A, S) &= |\{s_{io}^0 : s_{io}^0 = 1 \text{ and } q_i^{in,0} = q_{\max}^{in,0}\}| && \text{(most from longest input queues)}
\end{aligned}$$

where $q_{\max}^{in,t} = \max_i \{q_i^{in,t}\}$ is the length of the longest input queue t time steps in the future.

How do we interpret these objectives? Maximizing J_a corresponds to minimizing the average wait as follows. Consider the total wait of all packets in the queue at $t = f + 1$ if no packets are sent. A packet sent at time step t reduces the wait by $(f - t + 1)$. Similarly, J_a corresponds to minimizing the average queue size. Maximizing J_s maximizes the number of packets sent over the future periods. This criteria is not very useful since it does not discriminate when the packets are sent in the sequence. Sending packets now or later is equivalent as long as the total number sent is the same. $-J_q$ is the longest input queue size over the future periods. J_l is the number of packets sent from the longest input queues in the first time step [19, 25]. Maximizing J_q and J_l minimizes the required input buffer sizes.

Other objectives might include minimizing the longest wait of any packet over the future horizon or sending the most of the longest waiting packets. These objectives would prevent any packets from waiting too long, and are important when considering fairness. But, they are not considered here.

The objective used in this paper is based on the global criteria of minimizing the average wait time and the required buffer size. Preliminary studies indicate using $J = (J_a, J_q, J_l)$ with a lexicographic ordering. That is, first find a set of S that minimizes the average statistics. Among these, find the set of S that minimizes the longest queue over the future horizon. From these, choose an S that sends the most packets from the longest IQ.

4 Experiments

This section addresses the first question: *Does the future information make any significant performance improvements?* To see this, we simulate different size switches under different

loads using different future horizons, f . An ideal arbitrator searches for the best of the $(N!)^{f+1}$ possible send sequences (i.e. maximizes J defined in the previous section). In our experiments, an exhaustive search is used when $f > 0$. When $f = 0$, the matching algorithm in Appendix A.1 is used.

Experiments consist of an arrival process, normalized load, λ , switch size, N , and future horizon, f . Four representative processes generate arrivals: *uniform*, *output hot spot*, *input hot spot* and *bursty*. The uniform and output hot spot processes generate an arrival with probability λ in each time slot at each input. The uniform process distributes the destination uniformly across the outputs. The output hot spot process sends the fraction $\min\{1, 0.95/(\lambda N)\}$ of each input's arrivals to the first output, and distributes the remaining arrivals uniformly across the remaining outputs. If the total switch load (λN) is greater than 0.95, the first output has a load of 0.95. The remaining outputs have the remaining load.

The input hot spot process generates an arrival at the first input with probability $\lambda_1 = \min\{0.95, \lambda N\}$ and at the other inputs with probability $\frac{\lambda N - \lambda_1}{N-1}$. The destination is uniformly distributed across the outputs. If the total switch load is greater than 0.95, the first input has a load of 0.95 and the remaining inputs have the remaining load. For the bursty process, packets arrive in geometrically distributed bursts of at least one packet with mean burst size b . At any input, a burst starts in a time slot with probability λ/b . All packets in a burst arrive in consecutive time slots and have the same destination chosen uniformly. Bursts start immediately or as soon as all prior bursts finish, whichever is first. In this paper $b = 10$.

The IQ switch is simulated and statistics are collected for 10^7 time slots after an initial period of 5000 time slots to allow the queues to reach steady state. For a given arrival process, load, and switch size, the arrivals are identical across all experiments. Statistics include the mean wait time (the total queueing time minus the 1 time slot service time) and the queue-size distribution (i.e., F_B the probability that the queue is larger than B). In the case of the input hot spot, the IQ statistics are restricted to the overloaded input.

The following robust procedure provides estimate error bars. An experiment's data is

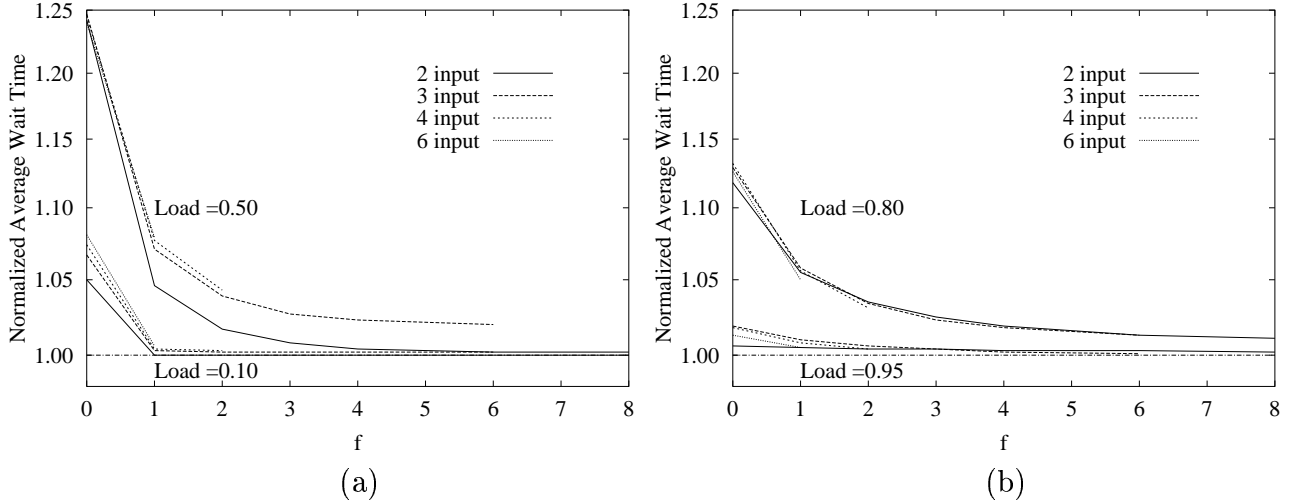


Figure 1: Plot of mean wait time for an input-queued switch as a function of the number of future look-ahead time steps, f , for different switch sizes under the uniform arrival process. All results are normalized by the wait time of the same size output-queued switch with the same arrival process.

divided into 10 consecutive periods and a separate estimate is made for each sub-period.² The standard deviation of these sub-estimates bounds the error. Across experiments, the mean wait time has a standard deviation of less than 0.7% (of the reported value) with a median value (across all data) of 0.1%. F_B has a standard deviation of less than 9% with a median value of 6% for $F_B > 10^{-3}$.

The experiments are repeated for an OQ shared buffer switch. The IQ switch wait times are normalized by the corresponding OQ switch results. By Little’s formula, the normalized wait time is also the normalized average queue size. The OQ switch shared buffer sizes are divided by N to get the buffer size per input port.

Figures 1–3 show the results. The $O((N!)^{f+1})$ time complexity with future information limits the switch sizes that could be simulated to $N \leq 6$. Without future information, switch sizes up to $N = 32$ are tested. We make several observations.

Figure 1 shows IQ wait times are as much as 25% larger than with OQ for the uniform arrival process without future information. But, with more future information the IQ perfor-

²The queuing times and buffer size have strong correlations over time. The sub-sequences chosen were large enough so that the sub-period estimates could be considered independent estimates.

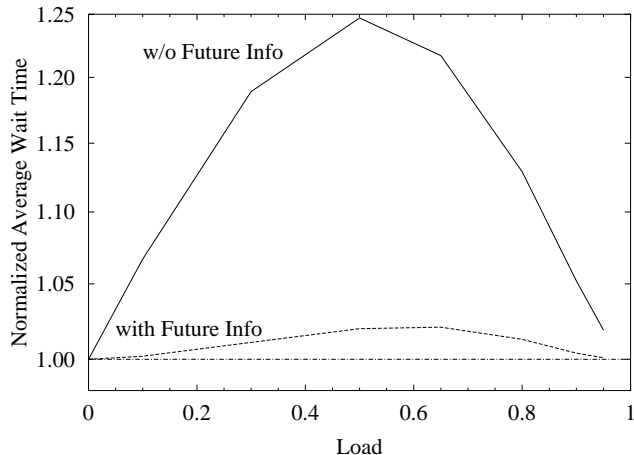


Figure 2: Plot of normalized wait time for an input-queued switch as a function of load without ($f = 0$) and with ($f = 6$) future information for an $N = 3$ switch under the uniform arrival process.

mance approaches the OQ performance. For $N = 2$, the IQ switch performance asymptotes to the OQ switch performance. The larger switches appear to asymptote at higher levels that are similar across the larger switch sizes.

Figure 2 plots delays with and without future information for $N = 3$ and the uniform arrival process. Based on the previous figure, $N = 3$ is representative of all $N \geq 2$ switches. At high and low loads the IQ and OQ switches have equally good performance with and without future information. The arrival constraint (1) limits conflicts within an arrival, a , to the output constraint (2). At low loads queueing is rare and caused by isolated conflicts within a . In this case, the output constraint affects both IQ and OQ switches equally. At high loads, the arrival constraint and the objective J_q tends to balance the many queueing packets among inputs. Instead, congestion tends to be concentrated on a few outputs which adversely affects both IQ and OQ switches equally. At intermediate loads, the queueing is dominated by packet conflicts over short sequences of arrivals. In this case, the future information reduces the wait times from over 20% to within 2% percent of optimal performance.

Figure 3 shows the buffer distribution. Among the different arrival processes, the IQ has buffer distribution (when $F_B < 10^{-1}$) nearly identical to OQ for uniform and output hot spot traffic, within one packet of OQ for input hot spot, and within 10% of OQ for bursty traffic.

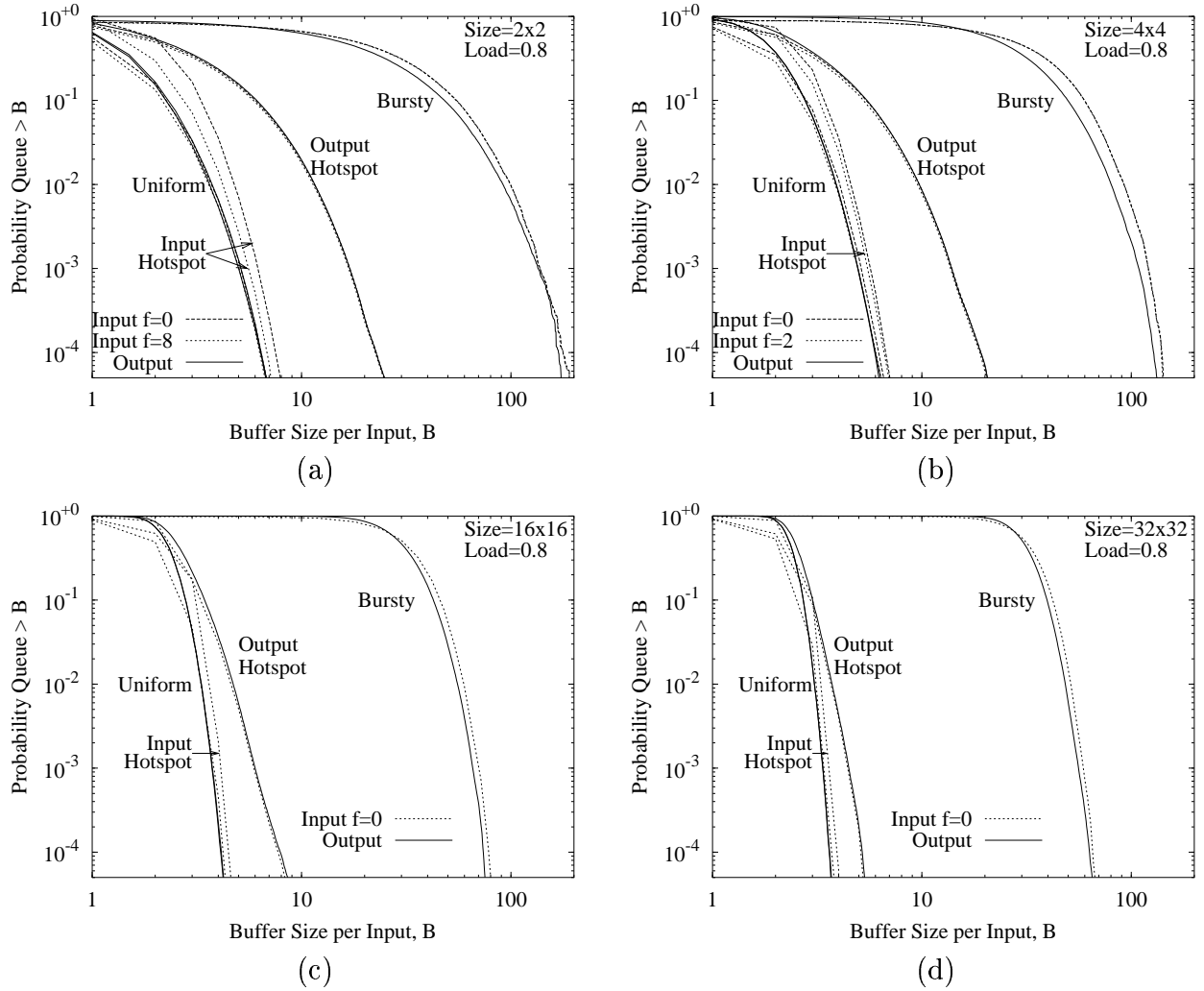


Figure 3: Queue-size distribution. The load is 0.8 and the number of inputs is 2 (a), 4 (b), 16 (c), and 32 (d)

We emphasize that these comparisons are between the shared buffers in the OQ switch and the less efficient segregated buffers of the IQ switch. The buffer performance with the IQ switch is dramatically better than with *non-shared* output buffering. For instance, in output hot spot traffic queued packets are concentrated on the overloaded output. With non-shared OQ, this requires a large buffer at every potential hot spot output. With IQ, the packets waiting for the overloaded output are distributed evenly across the input queues.

Surprisingly, the IQ switch is less affected by input hot spot traffic than output hot spot traffic. A simple example will show why. Suppose a 2×2 switch has an output hot spot

load of $\lambda = 0.5$ on each input and all packets are destined for one output. Whenever two packets arrive in the same time slot, packets will queue. Suppose, instead there is an input hot spot load of $\lambda = 1.0$ on one input, zero load on the other, and the arrivals are uniformly distributed across the outputs. Though the total load is the same as in the first case, in the latter the switch will send the one arrival in each time slot with no queueing.

Future information has little effect on the tail of the buffer size distribution. The no future information case used the Appendix A.1 matching algorithm to maximize J_a and J_l . Most of the performance gain in Figure 1 is achieved with $f = 1$. This suggests that an arbitrator that sends the maximum number of packets, takes care to send from the longest input queues, and looks only a few steps in the future can achieve most of the possible gains.

5 Equivalence between input and output queueing

In the previous section's experiments, IQ and OQ switches have similar performance with future information. Can the switches be equivalent? In the strict sense they would be equivalent if, under any arrival sequence, they send the same *packets* in each time slot. We consider a weaker notion in which they send the same *number of packets* in each time slot. We say two switches X and Y of the same size are *equivalent* if given any sequence of arrivals $A = \{a^t\}$, and starting with empty queues, switch X can send the same number of packets in every time slot as switch Y and vice versa.³

The notion of equivalence here includes probabilistic controllers as long as they always send the same number of packets. For instance, given two waiting packets from different inputs for the same output. In the next time slot a single packet arrives at one of these inputs for a different output. The OQ switch can always send all three packets in the two time slots. The IQ switch without future information may or may not be able to send all three packets depending on its packet choice in the first time slot and at which input the new packet

³More formally, let $S_X(A) = \{s_X^t\}$ be the sequence of packets sent by switch X given A and starting from empty queues. Let $\sigma_t(S_X(A)) = \sum_{i=1}^t |s_X^i|$. If for any A and $S_X(A)$, $\sigma_t(S_Y(A)) \geq \sigma_t(S_X(A)) \forall t$ then we write $Y \geq X$. X and Y are *equivalent* if $X \geq Y$ and $Y \geq X$.

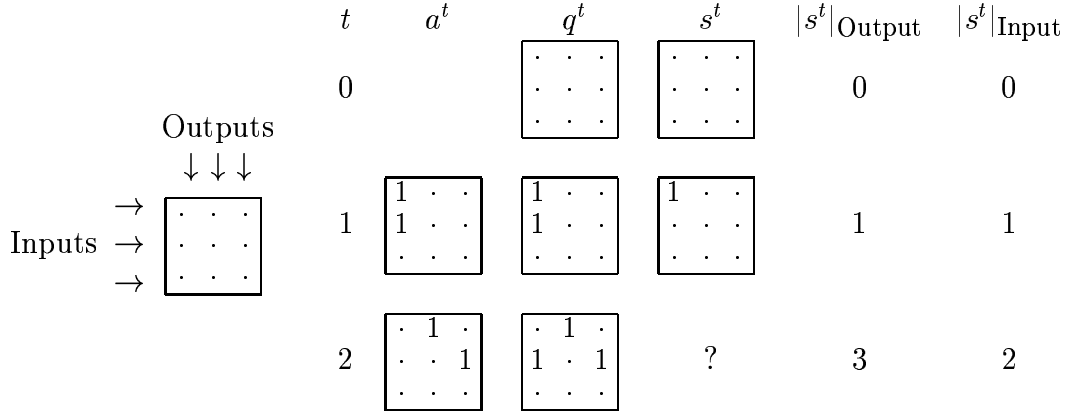


Figure 4: Set of arrivals showing input and output switches are not equivalent for $N \geq 3$ inputs. For each time step t , the new arrivals a^t , the queue state, q^t , the packets sent s^t , and the number of sent packets $|s^t|$ are shown. For clarity, zero packets are represented by “.”. At $t = 2$, the output-queued switch can send 3 packets and regardless of which packet the input-queued switch chose to send at $t = 1$, it can only send two packets at $t = 2$.

arrives. Thus IQ switches without future information and OQ switches are not equivalent. This section shows two results about the equivalence of IQ with future information and OQ switches and discusses the impact on performance.

First, IQ and OQ switches are not equivalent for $N \geq 3$. Figure 4 shows an example for $N = 3$ inputs where, regardless of the arbitration decisions, the IQ switch sends fewer packets than the OQ switch. Since this example can be embedded in any larger switch size, the result applies to $N \geq 3$. At $t = 0$, 2 packets destined for output 1 arrive. At $t = 1$, only one can be sent by either the IQ or OQ switch. Two packets arrive at the same two inputs destined for the second and third output. The OQ switch can send the 3 packets waiting at 3 outputs. The IQ switch has packets at only 2 inputs and can send 2 packets, no matter which packet it chose to send at $t = 1$.

For $N = 2$, IQ switches with future information and OQ switches are equivalent as stated in the following theorem proved in Appendix A.2.

Theorem 5.1 *A 2×2 IQ switch with future information is equivalent to a 2×2 OQ switch*

Larger IQ switches are strictly not equivalent to OQ switches, so performance is strictly worse. Since the 2×2 IQ and OQ switches send the same number of packets in every time

	$N = 2$	N bounded	N unbounded
$f = 0$	$O(N^{2.5})$		
f bounded	$O(1)$	$O(f^{N^2+1})$	NP-Hard
f unbounded	$O(1)^*$		

* This case is also equivalent to the OQ switch.

Table 1: Complexity of using future information vs. switch size, N , and future horizon, f .

slot, mean wait times and the total queue size distribution are identical. The wait-time distribution is not the same, since the two switches send different packets. The experiments show that enough freedom exists in the packet arbitration so that the segregated IQ size distribution is approximately $1/N$ the total shared-buffer OQ size distribution, leading to approximately the same buffer-size requirements for IQ and OQ switches.

6 Complexity of Using Future Information

This section gives complexity results for various conditions on N and f . The results are summarized in Table 1.

6.1 Optimally using future information is hard for unbounded N

Given a bipartite graph, $B = (V_0, V_1, E)$, a set M of edges is a *matching* if no two edges in M are incident to the same node⁴. A maximum matching has the largest number of edges. If nodes V_0 , nodes V_1 , and edges E correspond to switch inputs, switch outputs, and waiting packets, then the maximum matching has the obvious direct equivalence to maximizing the number of packets sent, J_s . Finding a maximum matching has $O(N^{2.5})$ time complexity [12].

Given the next f arrivals, which packets sent now maximize the total number of packets sent over the next $f + 1$ time slots. We define a *time-step matching*. Given a bipartite graph, B , with vertex sets V_0, V_1 and a sequence of edge sets A_0, A_1, \dots, A_f , where each vertex of V_0 is on at most one edge of each set A_i , $i > 0$. Form a matching M_0 on (V_0, V_1, A_0) . Remove the edges in M_0 , add the edges in A_1 , form a new matching M_1 , and continue the

⁴Throughout this paper we use the term “bipartite graph” to denote a bipartite multigraph, i.e., parallel edges are allowed. We use the notation (x, y) to refer to an edge between vertices x and y even though there may be several such edges; no ambiguity results from this in our application.

process to matching M_f . This has the obvious correspondence with the switching problem. Note that the sets of edges arriving in each time period respect the arrival constraint, i.e., at most one edge arrives at each vertex of V_0 . The total matching size is $m = \sum_{i=0}^f |M_i|$. The *maximum time-step matching problem* is, given $(V_0, V_1, \{A_i\})$, find the matching that maximizes m . This is equivalent to finding the S that maximizes J_s . The next theorem, proved in Appendix A.3, shows that this problem is not easy.

Theorem 6.1 *Finding a maximum time-step matching is NP-hard.*

The next corollary, proved in Appendix A.4, shows several variations of the problem are difficult.

Corollary 6.2 *The following problems are NP-hard:*

- (a) *finding a maximum time-step matching, even when the edges arriving in each time period form a matching plus at most one extra edge;*
- (b) *finding a time-step matching with minimum average statistics (maximize $J_a(q, A, S)$);*
- (c) *finding a time-step matching that minimizes the maximum input queue size (maximize $J_q(q, A, S)$);*
- (d) *finding a maximum time-step matching for $f = 1$;*
- (e) *finding a maximum time-step matching for $f = 3$ when each vertex of V_0 is on at most one edge of each set A_i , $i \geq 0$.*

Corollary 6.2(a) shows there is a sharp jump in complexity in time-step matching: If A_i corresponds to a matching then every packet in A_i could be trivially sent immediately without conflict. But relaxing this assumption by even one edge makes the problem NP-hard. Corollaries (b) and (c) indicate that changing the objective will not simplify the problem. Corollary (d) shows the problem becomes hard with even one time step of future information. Corollary (e) indicates the problem is hard if in the future arrivals, each output receives at most one packet, i.e., one set of conflicting arrivals now can not be resolved easily even if the future arrivals correspond to a matching. These variants are included to show the problem

difficulty does not follow from a particular narrow definition. Together these results show that if N grows unbounded, optimally using the future information is hard.

6.2 Finding an optimal set of packets is polynomial for fixed N

What is the complexity if N is fixed and f is allowed to grow unbounded? If the objective can be stated in the recursive form:

$$\begin{aligned} J(q, A, S) &= h(q, A, S, f + 1) \\ h(q, A, S, t + 1) &= g(q^t, s^t, a^{t+1}, t, h(q, A, S, t)) \\ h(q, A, S, 0) &= 0 \end{aligned} \tag{5}$$

for some $O(1)$ (given N is fixed) monotonic in h function g , then there is a polynomial algorithm that maximizes $J(q, A, S)$. All Section 3 objectives can be formulated as (5). For such objectives, Appendix A.5 proves the following theorem.

Theorem 6.3 *For fixed N and $J(q, A, S)$ of the form (5), finding an S given (q, A) that maximizes $J(q, A, S)$ has time complexity polynomial in f .*

Although this shows a polynomial solution for fixed N , the polynomial degree and the constants are large, (i.e. $O(f^{N^2+1})$) so it is not practical. Its main contribution is to show that it is polynomial and further algorithm work could produce a simpler algorithm.

6.3 Finding an optimal set of packets is $O(1)$ for $N = 2$

Consider a 2-input IQ switch. Packets sent now can affect whether the IQ switch can send the same number of packets as the OQ switch in the future. How far in the future? In Appendix A.6 we show these dependencies can last arbitrarily far into the future.

What if, instead, an entire arrival sequence of length T is available? The *2-input packet arbitration* problem is, given an empty IQ switch and the next T future arrivals, $A = (a^1, \dots, a^T)$. Compute the sequence of permutations, $P = (p^1, \dots, p^T)$, used to send packets so that the IQ and OQ switches are equivalent. Appendix A.6 proves the following.

Theorem 6.4 *The 2-input packet arbitration problem has an $O(T)$ time-complexity solution.*

In other words, over the T time slots the complexity is, on average, $O(1)$ per time slot.

How much future information is needed to resolve any given time slot? In simulations, a uniform load of $\lambda = 0.8$ requires $f = 2.3$ on average and $f = 22$ at the 99th percentile. At $\lambda = 0.6$ the average and 99th percentile are 0.53 and 7.

For large switches, using future information is hard. For $N = 2$ switches, an arbitrarily large f may be necessary to resolve optimally the next send. The next section explores a heuristic.

7 A simple search heuristic

Let $\mu(q)$ be the maximum number of packets that could be sent from queues in state q . Given q and A , a *maximum sequence*, S , is a set of packets sent s.t. $\forall s^t \in S, |s^t| = \mu(q^t)$. Instead of searching through all $(N!)^{f+1}$ possible S , this section analyzes the heuristic of only searching through maximum sequences. We present two results that depend on N .

First, maximum sequences are optimal for $N = 2$ as shown in the following theorem proved in Appendix A.7.

Theorem 7.1 *For $N = 2$ IQ switches, there are no (q, A, S) s.t.*

- (a) S is non-maximum and
- (b) $J_a(q, A, S) > J_a(q, A, S')$, or $J_s(q, A, S) > J_s(q, A, S')$ for all maximum sequences S' .

Though optimal for $N = 2$, the following theorem shows maximum sequences are not optimal for $N \geq 3$. Recall that maximizing J_a and J_s minimizes the average statistics and maximizes the number of packets sent.

Theorem 7.2 *For $N \geq 3$ IQ switches, there exist (q, A, S) s.t.*

- (a) S is non-maximum and
- (b) $J_a(q, A, S) > J_a(q, A, S')$, and $J_s(q, A, S) > J_s(q, A, S') \forall$ maximum sequences S' .

In other words, we have the counterintuitive result that to minimize the average delay (and queue size) through an $N \geq 3$ IQ switch, some time slots will send fewer than the maximum possible number of packets.

t	a^t	q^t	s^t	$ s^t $	w^t	t	a^t	q^t	s^t	$ s^t $	w^t
0		1 . 1 . 1 . . . 1	. . 1 . 1	2	8	0		1 . 1 . 1 . . . 1	1 . . . 1 . . . 1	3	12
1 1 . . .	1 1 . . 1	1 1 . . .	2	6	1 1 1 . . 1 1	1	3
2	1 . . . 1	1 . . . 1 . . . 1	1 . . . 1 . . . 1	3	6	2	1 . . . 1	1 . . . 1 1 . . .	1 1 . . .	2	4
3	. 1 . 1 1 . 1 1 . 1	2	2	3	. 1 . 1 1 . 1 1 1 . 1	2	2
			Total	9	22				Total	8	21

(a)
(b)

Figure 5: Example for $f = 3$ where taking fewer than the maximum number of packets in step zero (a) leads to a greater number of packets sent and lower average wait time than when the maximum number of packets is sent (b). For each time step t , the new arrivals a^t , the queue state, q^t , the packets sent s^t , the number of sent packets $|s^t|$, and the saved wait time $w^t = |s^t|(f - t + 1)$ are shown.

Proof: Figure 5 shows an example for a 3×3 switch. In Figure 5a only 2 packets are sent at $t = 0$, while in Figure 5b 3 packets are sent. No matter what choice is made at $t \geq 1$ in Figure 5b, only 2 input queues are non-empty at $t = 2, 3$ so that at most 2 packets can be sent in these steps. Both the total number of packets sent and the total saved wait time is larger in Figure 5a than Figure 5b. Since this example can be embedded into larger switch sizes, it applies for every $N \geq 3$. \square

How often do such situations occur? A monitor checked how often $s < \mu(q)$ in the experiments of Section 4. The maximum observed rate in the experiments was less than 1 in 300 time slots at a load of $\lambda = 0.8$. This rate decreased quickly with smaller or larger loads and increased slightly with increasing N or f . It was never observed for $f \leq 1$ or $N = 2$. The low rate of non-maximum sequences suggests the heuristic should be considered.

8 Conclusions

Much recent work has explored the limits of IQ switches. This paper explores using future arrival information on IQ switches with no switch fabric speed up. It answers three questions.

Can a perfect predictor improve IQ switch performance? This paper demonstrates that future information significantly improves IQ switch performance. It reduces mean delays and queue sizes by over 20% so that IQ and OQ switch wait-times are within 2%. Buffer-size performance with and without future information is similar. The experiments suggest that predictions over a short future horizon bring the majority of the performance gains.

Are IQ and OQ switches equivalent with future information? For 2×2 switches IQ and OQ can send the same number of packets when the IQ switch has future information for packet arbitration. An $O(1)$ time complexity per time slot algorithm can achieve this. Although IQ and OQ performance is similar, they are strictly not equivalent for larger switches.

What is the computational complexity of optimally using the future information? The complexity of optimally using the future information is NP-hard when the size of the switch is unbounded. With bounded switch size, a polynomial algorithm (but, with large degree) is given. One simplifying heuristic only sends packet sets with the maximum number of packets in each time slot. For 2×2 switches this is optimal. Counter intuitively for larger switches, sending less than the maximum number of packets may be necessary to optimize future performance. This situation rarely occurs in simulations, suggesting that always sending the maximum number of packets would be a valid heuristic.

The dichotomy between 2×2 and larger switches can be traced to the fact that only one permutation can send a given packet in the 2×2 switch. Since the permutation that sends a packet is known, the only degree of freedom is when to send the packet. In larger switches, many possible permutations (i.e. $(N - 1)!$) can send a given packet.

Future information may have the greatest impact in optics. Optics has proposed explicit mechanisms for providing future arrival information. Typical optical systems operate at relatively low loads (e.g. 30–60%). This range of loads is precisely where future information

can have the greatest effect (see Figure 2). At such low loads only a few slots are sufficient for nearly all the possible gains. Empirically, the time of the exhaustive search in the experiments is much less with low loads suggesting that computational costs are less in this regime. More importantly, buffering in the optical domain requires expensive and limited fiber delay lines. So, any mechanism for reducing contention has some value.

Future work will look at simplifying approximations that approach the theoretical performance limits presented here. Issues such as fairness and quality of service also need to be investigated. Specific architectures that generate the future information need to be developed. Whether or not future information proves practical, this paper frames what is possible.

Appendices

A.1 $O(N^{2.5})$ arbitrator when $f = 0$

Without future information ($f = 0$), $J_s = J_a$ and if S maximizes J_l , then it maximizes J_q . This section presents a polynomial algorithm that simultaneously maximizes J_l and J_s (and thus $J = (J_a, J_q, J_l)$) based on maximum matching algorithms. This problem could also be solved with a weighted matching of $O(N^3 \log N)$, but, the solution here is less complex.

Recall that J_l is the number of packets sent from the longest input queues. Let $|M'|^*$ be the most packets that can be sent from the longest input queues. Let $|M|^*$ be the most packets that can be sent from all queues.

Theorem 1.1 *A polynomial-time matching algorithm sends both $|M'|^*$ packets from the longest input queues and $|M|^*$ packets from all input queues.*

Proof: Refer to Section 6.1 for notation. The polynomial-time algorithm in [12] starts with a possibly empty initial feasible matching followed by a series of *augmentation* steps, as shown in Figure 6. Once a node in V_0 or V_1 is incident to a matching, augmentation may change the edges in the matching but does not remove any incident nodes. Given that V_0 corresponds to inputs, this suggests the following algorithm. Let $V'_0 \subseteq V_0$ correspond to the inputs with

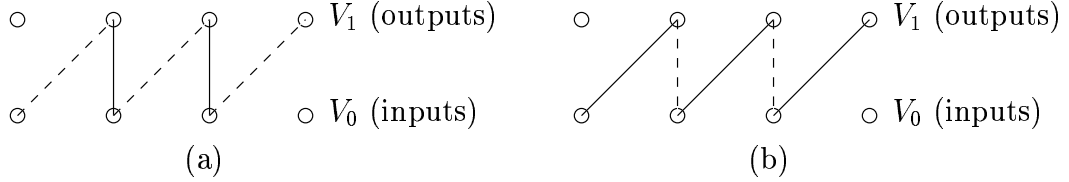


Figure 6: The matching in (a) with 2 of the 5 edges (indicated by the solid lines) is augmented in (b) to contain 3 of the five edges.

the longest queues, i.e. $V'_0 = \{i | q_i^{in} = q_{\max}^{in}\}$. Let $E' \subseteq E$ be the edge subset induced by V'_0 . Create a maximum matching, M' on $B' = (V'_0, V_1, E')$. M' in turn serves as an initial feasible matching for B , which results in matching M . M' contains the maximum number of edges incident on V'_0 (i.e. maximum number of packets from the longest input queues). M contains the maximum matching, and by the conservation of nodes in the augmentation step, M is also incident to the maximum number of nodes in V'_0 . The algorithm is at most twice the time of the original polynomial-time algorithm. \square

A.2 Input and output-queued switches are equivalent for $N = 2$

This appendix proves Theorem 5.1.

Proof: The OQ switch is not subject to the input constraint and can always send at least as many packets as the IQ switch. A 2×2 IQ switch has only two permutations, p , in (4). We prove the IQ switch with future information can choose a p in each time slot that always sends as many packets as the OQ switch. Thus, it meets our definition of equivalence.

Consider the case where both inputs receive a packet for output o (denoted an *o-choice*). Given such an arrival, an IQ or OQ switch could choose to send either packet—but not both. Either choice is the same for the OQ switch since either choice would result in one more packet waiting at output o . For the IQ switch the choice affects which input will have an additional packet waiting. The choice is left pending during the inspection of future arrival information. Let c_o be the number of pending *o*-choices and denote the *choice state* by (c_1, c_2) . The choice state is an auxiliary description of the underlying queue state that facilitates the resolution of pending choices.

In the OQ switch, c_o is simply the number of packets queueing at output o . Depending on how the pending choices are resolved in the IQ switch, the c_o packets waiting for output o can be distributed in any manner between input 1 or input 2. The following lemma shows that the choice state is a sufficient description to resolve every pending choice such that the IQ and OQ switches send an equal number of packets in every time slot.

Lemma 1.2 *Given a 2×2 IQ switch whose state just before the arrivals can be described by the choice-state (c_1, c_2) . For any set of packet arrivals, a set of packets can be sent so that the new switch state can also be described by a choice state and the number of packets sent from each output is the same as an OQ switch in the same state.*

Proof: Only 9 different arrivals satisfy the arrival constraint (1). If o denotes one OQ, let \bar{o} denote the other queue. If p denotes one of the two permutations, let \bar{p} denote the other permutation. Table 2 shows the transitions for each combination of arrival and (c_1, c_2) . Table 3 describes the transition rules that generated the table. These rules select which permutation is used to send packets so that the choice-state representation is always maintained.

Given (c_1, c_2) , output o has c_o packets waiting. The choice-state is the state after a send and before an arrival. Thus with arrival a , an OQ switch in the same state would send:

$$|s| = \begin{cases} 0 & \text{if } c_o = a_{io} = 0, i = 1, 2, o = 1, 2 \\ 2 & \text{if } c_o + a_{1o} + a_{2o} > 0, o = 1, 2 \\ 1 & \text{otherwise} \end{cases} \quad (6)$$

An inspection of Table 2 shows the IQ switch with the choice-state representation sends the same number of packets as (6) in every case. \triangle

An empty switch has choice-state $(0, 0)$. By the above lemma, with each arrival the choice-state can be recomputed so that the number of packets sent by the IQ switch is equal to the number sent by the OQ switch. Thus, the IQ and OQ switches are equivalent. \square

A.3 Finding a maximum time-step matching is NP-hard.

This appendix proves Theorem 6.1.

Table 2: The choice-state transitions as a function of the arrivals in a 2-input input-queued switch. The initial state is (c_1, c_2) .

Case	Arrival	Next State	Type	Num Sent
1	$\begin{bmatrix} \cdot & \cdot \\ \cdot & \cdot \end{bmatrix}$	if $c_o > 0, o = 1, 2$ $(c_1 - 1, c_2 - 1)$ else if $c_1 > 0, c_2 = 0$ $(c_1 - 1, 0)$ else if $c_1 = 0, c_2 > 0$ $(0, c_2 - 1)$ else $(0, 0)$	1 1 1 0	2 1 1 0
2	$\begin{bmatrix} 1 & \cdot \\ \cdot & 1 \end{bmatrix}$	(c_1, c_2)	2	2
3	$\begin{bmatrix} \cdot & 1 \\ \cdot & \cdot \end{bmatrix}$	if $c_1 > 0$ $(c_1 - 1, c_2)$ else $(0, c_2)$	3 3	2 1
4	$\begin{bmatrix} 1 & \cdot \\ \cdot & \cdot \end{bmatrix}$	if $c_2 > 0$ $(c_1, c_2 - 1)$ else $(c_1, 0)$	3 3	2 1
5	$\begin{bmatrix} 1 & \cdot \\ 1 & \cdot \end{bmatrix}$	if $c_2 > 0$ $(c_1 + 1, c_2 - 1)$ else $(c_1 + 1, 0)$	4 4	2 1
6	$\begin{bmatrix} \cdot & 1 \\ \cdot & 1 \end{bmatrix}$	if $c_1 > 0$ $(c_1 - 1, c_2 + 1)$ else $(0, c_2 + 1)$	4 4	2 1

Table 3: The types of transitions and the permutations, p , selected in Table 2.

Send Type	Comment
0	The queues are empty. No packets to send.
1	Take one o -choice if $c_o > 0$ for each $o = 1, 2$. With no arrivals to force a choice, select p arbitrarily. Assign \bar{p} to the time slot where each o -choice was generated.
2	Select the p that immediately sends the arriving packet(s). Since both arrivals can only be sent by p , two is the maximum number of packets that the switch can send, and therefore sending p now maximizes the number of packets sent.
3	With only one arrival for output o , select the p that sends the arrival now. If $c_{\bar{o}} > 0$, take one \bar{o} -choice and assign \bar{p} to the time slot where the \bar{o} -choice was generated.
4	The two arrivals for output o generate an o -choice. The p will be determined in a later time slot. If $c_{\bar{o}} > 0$, take one \bar{o} -choice and, when p is determined, assign \bar{p} in the time slot where the \bar{o} -choice was generated. This creates a linked list of pending assignments.

Proof: In the *Disjoint Path Problem* (DPP) we are given a digraph along with k ordered pairs of vertices (s^i, t^i) , $i = 0, \dots, k-1$. Here k can be an arbitrary value. We wish to decide if there is a collection of edge-disjoint paths, the i th path of which goes from s^i to t^i . This problem was proved NP-complete in [7]. We shall refer to the vertices s^i, t^i as “terminals”, and (s^i, t^i) as a “terminal pair.”

We reduce DPP to time-step matching as follows. Consider an instance of DPP specified by digraph $G = (V, E)$ and terminals (s^i, t^i) , $i = 0, \dots, k-1$. Let n be the number of vertices ($n = |V|$) and let d be the maximum out-degree of a vertex. Partition E into d disjoint sets E_j , $j = 0, \dots, d-1$ such that in each E_j , each vertex has out-degree at most one. (Any partition satisfying this constraint will do.)

Let $f = d + k - 1$ in a time-step matching problem. Figure 7 shows an example. In general, the vertex sets of B are defined by:

$$V_i = \{v_i, \bar{v}_i : v \in V\} \cup \{u_i\}, \quad i = 0, 1.$$

Thus, $|V_i| = 2n + 1$. The edge sets are defined by:

$$\begin{aligned} A_j &= \{(\bar{v}_0, v_1) : v \in V\} \cup \{(v_0, w_1) : (v, w) \in E_j\}, & j = 0, \dots, d-1; \\ A_{d+j} &= \{(\bar{v}_0, \bar{v}_1) : v \in V\} \cup \{(v_0, v_1) : v \in V - \{s^j, t^j\}\} \\ &\quad \cup \{(u_0, s_1^j), (t_0^j, u_1)\}, & j = 0, \dots, k-1. \end{aligned}$$

We claim G contains the desired paths of DPP if and only if B has a time-step matching containing at least $\ell = dn + k(2n + 1)$ edges. The claim establishes the theorem. We now prove both directions of the claim.

Suppose G contains the desired edge-disjoint paths, say paths P_i from s^i to t^i , $i = 0, \dots, k-1$. In each of the first d time periods, match every edge (\bar{v}_0, v_1) for $v \in V$ that arrives in that period. In the remaining time periods $d + i$, $i = 0, \dots, k-1$, match every edge (\bar{v}_0, \bar{v}_1) for $v \in V$ that arrives in that period. In addition, match each edge (v_0, w_1) for $(v, w) \in P_i$, (v_0, v_1) for $v \in V - P_i$, and $(u_0, s_1^i), (t_0^i, u_1)$. (Edges (v_0, w_1) arrived in one of the first d periods, and the other edges arrived in period $d + i$.)

It is easy to see that we have specified a matching in each time period and that each edge is used at most once after it arrives. The number of matched edges is n in each of the first d periods and $2n + 1$ in the remaining periods, for a total of ℓ edges as desired.

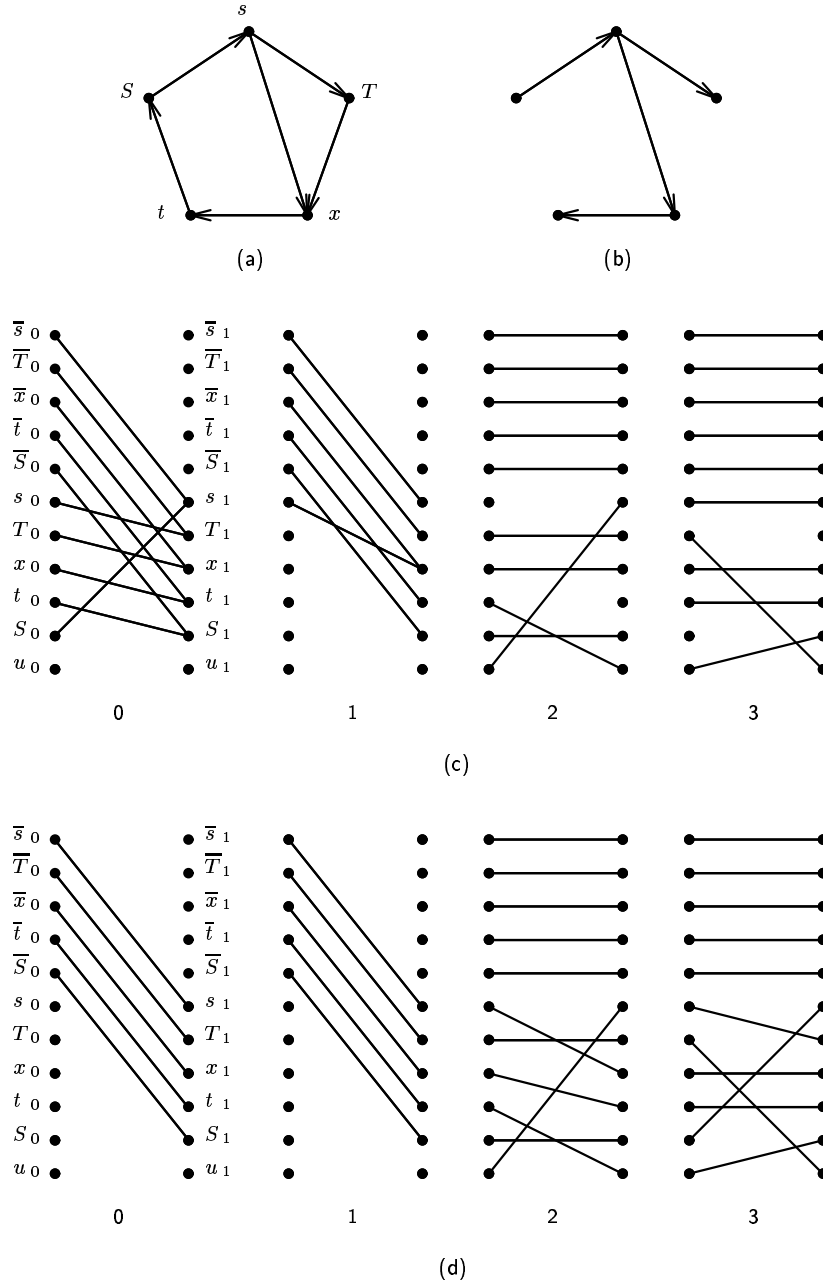


Figure 7: Example of the NP-hardness reduction. (a) DPP instance with $s^0 = s, t^0 = t, s^1 = S, t^1 = T$. (b) DPP solution. (c) Corresponding time-step matching instance. (d) Corresponding time-step matching solution.

Next, suppose B contains a time-step matching with at least ℓ edges. At most n edges are matched in each of the first d periods, since every edge arriving in these periods is incident to a vertex $v_1, v \in V$. At most $|V_1| = 2n + 1$ edges are matched in each of the remaining k periods. Thus any time-step matching contains at most ℓ edges. We conclude that the given

time-step matching contains exactly n edges in each of the first d periods and $2n + 1$ edges in each of the remaining k periods.

We have shown the time-step matching contains a perfect matching in the last k periods. The entire edge set $\cup A_i$ contains only k edges incident to each of the vertices \bar{v}_1 , $v \in V$, and u_0, u_1 . Hence these edges must be matched in the last k periods. In particular edges (\bar{v}_0, \bar{v}_1) for $v \in V$ are matched in each of the last k periods. The remaining edges form a perfect matching on the sets $\{v_i : v \in V\} \cup \{u_i\}$, $i = 0, 1$.

We have shown edges (u_0, s_1^i) and (t_0^i, u_1) are matched in period i , $i = d, \dots, d + k - 1$. Hence in each of the periods $d, \dots, d + k - 1$, the matched edges of the form (v_0, w_1) where $v \neq w$ correspond to edges (v, w) in G that form a path from s^i to t^i , plus zero or more cycles. All of these k paths in G are edge-disjoint since any given edge of B occurs in at most one of the perfect matchings. This gives the desired solution to DPP. \square

A.4 Maximum time-step matching variants are NP-hard.

This appendix proves Corollary 6.2

Proof: Each result is proven by modifying the reduction of the theorem. For each part we state the modification. The modified reductions are proven correct by arguments similar to Theorem 6.1. We leave the details of those arguments as simple exercises.

(a) The edges arriving in each of the last k periods form a matching. So we only need to modify the sets of edges arriving in the preceding periods. Redefine the sets E_j so that each is a singleton, i.e., write E as the disjoint union of $|E|$ sets E_j . Also, change the value of d to $|E|$. Now the same definition of A_j for $j = 0, \dots, d - 1$ makes each A_j a matching plus one extra edge (the edge of E_j).

(b) In Section 3, minimizing the average wait time is equivalent to maximizing $\sum_{t=0}^f (f - t + 1)n_t$, for n_t equal to the number of edges matched in period t . Theorem 6.1 proves that in the time-step matching problem of the reduction, any matching has $n_t \leq n$ for $t < d$ and $n_t \leq 2n + 1$ for $d \leq t < d + k$. Furthermore, equality holds if and only if the matching contains ℓ edges. Now it is easy to see the original reduction proves (b).

(c) We prove that two restricted versions of DPP are NP-complete, and then prove (c). We first prove DPP is NP-complete when restricted to instances where the maximum out-degree of a vertex is exactly equal to the number of terminal pairs, i.e., $d = k$. We do this by showing how to change an arbitrary instance of DPP into one with $d = k$ without affecting the solution. If the given instance has $d < k$, add two new vertices x, y with k new edges (x, y) . If the given instance has $d > k$, add two new vertices x, y with $d - k$ new edges (x, y) , and $d - k$ new terminal pairs (x, y) . In both cases the new vertices and edges have no affect on the other paths of the solution.

Next we prove DPP is NP-complete when every vertex has out-degree exactly k . To do this, start with an instance of DPP where $d = k$. Add two new vertices x, y with d new edges (x, y) and d new edges (y, x) . Furthermore, for every original vertex v with out-degree $d_v < d$, add $d - d_v$ new edges (v, x) .

Now we prove (c) by modifying the reduction of Theorem 6.1. Assume that in the given instance of DPP, every vertex has out-degree $d = k$. Add three vertices x_i, y_i, z_i to V_i , $i = 0, 1$, to the corresponding time-step matching problem. We also use some additional time periods, which, for consistency with the theorem, we index as $-d, \dots, -1$ for a total of $3d$ periods indexed by the consecutive integers from $-d$ to $2d - 1$. Define the edge sets:

$$\begin{aligned}
A_j &= \{(\bar{v}_0, z_1) : v \in V\} \cup \{(u_0, z_1), (x_0, z_1), (y_0, z_1)\}, & j = -d, \dots, -1; \\
A_j &= \{(\bar{v}_0, v_1) : v \in V\} \cup \{(v_0, w_1) : (v, w) \in E_j\} \\
&\quad \cup \{(x_0, z_1), (y_0, z_1), (z_0, z_1)\}, & j = 0, \dots, d - 1; \\
A_{d+j} &= \{(\bar{v}_0, \bar{v}_1) : v \in V\} \cup \{(v_0, v_1) : v \in V - \{t^j\}\} \\
&\quad \cup \{(u_0, s_1^j), (t_0^j, u_1), (z_0, z_1)\}, & j = 0, \dots, d - 1.
\end{aligned}$$

In contrast with Theorem 6.1, edge (s_0^j, s_1^j) arrives in period $d + j$, $j = 0, \dots, d - 1$. This ensures that an edge arrives at each vertex $v_0, v \in V$ in those periods.

We claim the DPP instance has the desired paths if and only if there is a time-step matching where no input queue ever contains more than d edges (in any time period).

Suppose the paths of the DPP instance exist. Construct the time-step matching as in Theorem 6.1 with these additions: match edges (x_0, z_1) in each of the first d periods, (y_0, z_1) in each of the next d periods, and (z_0, z_1) in each of the last d periods. It is easy to see that

for the resulting time-step matching, no IQ ever contains more than d edges. (In fact, every queue contains exactly d edges at the end of period $2d - 1$.)

Next suppose there is a time-step matching where no IQ ever contains more than d edges. Since each of the edges (x_0, z_1) , (y_0, z_1) , (z_0, z_1) arrives $2d$ times, d copies of each must be in the time-step matching. Thus, no edge (\bar{v}_0, z_1) , (u_0, z_1) is ever matched. This implies each vertex \bar{v}_0 , $v \in V$ has a queue of d edges in every nonnegative time period. Thus, each edge (\bar{v}_0, v_1) , (\bar{v}_0, \bar{v}_1) is matched in the period of its arrival. Hence in period d , each vertex v_0 , $v \in V$ has a queue of d edges. It now follows that in each of the last d periods, each vertex v_0 , $v \in V$ and u_0 is incident to an edge that is matched in that period.

It is easy to see that in period $d + j$, $j = 0, \dots, d - 1$, (u_0, s_1^j) is matched. Now we show that (t_0^j, u_1) is also matched in that period. The n vertices v_0 , $v \in V$ are adjacent to only $n + 1$ vertices, v_1 , $v \in V$ and u_1 . Since (u_0, s_1^j) is matched, an edge incident to u_1 must be matched. The only such edge is (t_0^j, u_1) . Hence (t_0^j, u_1) is matched.

The rest of the argument follows the theorem, showing the edges matched in the last d time periods give the desired paths that solve DPP.

(d)–(e) The argument is based on the NP-completeness of what we denote *restricted DPP* (RDPP), defined as follows: We are given a digraph G along with four disjoint sets of vertices S_i, T_i , $i = 1, 2$. We wish to decide if there is a collection of edge-disjoint paths such that for $i = 1, 2$ and each $t \in T_i$, there is a path from some vertex of S_i to t . Equivalently we seek two collections of paths \mathcal{P}_i , $i = 1, 2$ such that any edge of G belongs to at most one path of $\mathcal{P}_1 \cup \mathcal{P}_2$, each path of \mathcal{P}_i goes from a vertex of S_i to a vertex of T_i , and each vertex of T_i is the end of a path of \mathcal{P}_i . In addition the following restrictions hold:

(i) For $i = 1, 2$, each vertex of S_i has in-degree 0 and out-degree 1, and each vertex of T_i has out-degree 0 and in-degree 1.

(ii) Each vertex of G has out-degree ≤ 2 .

(iii) $|S_1| = |T_1| = 1$;

(iv) The paths of \mathcal{P}_2 are vertex-disjoint.

By (iii) let σ (τ) be the unique vertex of S_1 (T_1).

We will show that the NP-hardness of RDPP implies (d) and (e). RDPP itself can be proved NP-hard using a slight modification of the argument of [7]. The main change is to use the set S_2 to model vertex s_2 of [7]. Details are left as an exercise.

We first show (e). As in the Theorem 6.1 proof, we reduce RDPP to the time-step matching problem of (e). Consider a RDPP instance specified as above. We make two simplifying assumptions: The edge from σ does not go to T_2 , and no vertex has 2 edges directed to T_2 . These assumptions are warranted since otherwise the RDPP instance has no solution.

Let ST be the set of all terminals, $ST = \cup_{i=1}^2 S_i \cup T_i$. Partition the edges of $G - ST$ (i.e., the edges that are not incident to a terminal) into disjoint sets E_j , $j = 0, 1$ such that in each E_j , each vertex has out-degree at most one. This can be done by (ii).

Our time-step matching problem has $f = 3$. The vertex sets of B are defined by:

$$V_i = \{v_i, \bar{v}_i : v \in V\}, \quad i = 0, 1.$$

The edge sets are defined by:

$$\begin{aligned} A_j &= \{(\bar{v}_0, v_1) : v \in V - ST\} \cup \{(v_0, w_1) : (v, w) \in E_j\}, & j = 0, 1; \\ A_{2+j} &= \{(\bar{v}_0, \bar{v}_1) : v \in V - ST\} \\ &\cup \{(v_0, v_1) : v \in V - ST, \text{ no edge from } v \text{ to } T_j \text{ exists}\} \\ &\cup \{(v_0, w_1) : (v, w) \in E, v \in S_j \text{ or } w \in T_j\} \\ &\cup \{(\sigma, \tau) : j = 1\}, & j = 0, 1. \end{aligned}$$

These arrival sets satisfy the hypothesis of (e) (use (i) and the simplifying assumptions here).

Let n_0 be the number of nonterminals, $n_0 = |V - ST|$. We claim G contains the desired paths of RDPP if and only if B has a time-step matching containing at least $\ell = 6n_0 + |T_2| + 2$ edges. The claim establishes the theorem. We now prove both directions of the claim.

Suppose G contains the desired paths, say paths P from σ to τ and Q_i from s_i to t_i , $i = 1, \dots, |T_2|$ (where $T_2 = \{t_i : i = 1, \dots, |T_2|\}$ and each $s_i \in S_2$). In each of the first 2 time periods, match every edge (\bar{v}_0, v_1) arriving in that period. In the remaining 2 time periods match every edge (\bar{v}_0, \bar{v}_1) arriving in that period. In addition in time period 2 match each edge (v_0, w_1) for $(v, w) \in P$ and each edge (v_0, v_1) for $v \in V - ST - P$. Similarly in time period 3 match each edge (v_0, w_1) for $(v, w) \in \cup Q_i$, and each edge (v_0, v_1) for $v \in V - ST - \cup Q_i$.

Also match edge (σ, τ) . (Note that the first and last edges of P arrive in period 2; the first and last edges of each Q_i arrive in period 3; the remaining edges of all paths arrive in one of the first 2 periods, and the other matched edges arrive in the same period they are matched.)

Property (iv) implies the edges in period 3 form a valid matching. It is obvious that we have used each edge at most once, after it arrives. So we have a valid time-step matching. The number of matched edges is n_0 in each of the first 2 periods, $2n_0 + 1$ in period 2 and $2n_0 + 1 + |T_2|$ in period 3. This gives ℓ matched edges as desired.

For the opposite direction suppose B contains a time-step matching with at least ℓ edges. At most n_0 edges are matched in each of the first 2 periods, since every edge arriving in these periods is incident to a vertex $v_1, v \in V - ST$. At most $2n_0 + 1$ edges are matched in period 2 since every edge that has arrived is incident to τ_1 or a vertex $v_1, \bar{v}_1, v \in V - ST$. Similarly at most $2n_0 + |T_2| + 1$ edges are matched in period 3. Hence any time-step matching contains at most ℓ edges. We conclude that the given time-step matching has cardinality exactly $n_0, n_0, 2n_0 + 1$ and $2n_0 + 1 + |T_2|$ in periods 0 through 3 respectively.

We have shown the time-step matching contains a matching covering every vertex $v_1, \bar{v}_1, v \in V - ST$ in the last 2 periods. The entire edge set $\cup A_i$ contains only 2 edges incident to each vertex $\bar{v}_1, v \in V$. Hence the edges (\bar{v}_0, \bar{v}_1) for $v \in V$ are matched in each of the last 2 periods. The remaining edges form a matching covering the sets $\{v_1 : v \in V - ST \cup T_1\}$ in period 2 and $\{v_1 : v \in V - ST \cup T_1 \cup T_2\}$ in period 3.

In period 2, the matched edges of the form $(v_0, w_1), v \neq w$ correspond to edges (v, w) in G that form a path from σ to τ , plus zero or more cycles. (Use (i) here.) In period 3, edge (σ, τ) is matched, since the only other edge incident to τ was matched in period 2. (Use (i) here.) Hence the matched edges of the form $(v_0, w_1), v \neq w$ correspond to edges (v, w) in G that form $|T_2|$ paths, each starting at a vertex of S_2 and ending at a vertex of T_2 , plus zero or more cycles. The paths are vertex disjoint (since they come from a matching) so (iv) holds. All of the paths in G are edge-disjoint since any given edge of B occurs in at most one of the matchings. This gives the desired solution to RDPP.

Part (d) is proved in a similar manner. The idea is to combine the first 3 time periods

defined above into one. This also allows us to discard the \bar{v}_i vertices. Specifically, the new bipartite graph B has vertex sets $V_i = \{v_i : v \in V\}$, $i = 0, 1$. The edge sets are:

$$\begin{aligned}
A_j = & \{(v_0, w_1) : (v, w) \text{ an edge of } G - ST, j = 0\} \\
& \cup \{(v_0, v_1) : v \in V - ST, \text{ no edge from } v \text{ to } T_j \text{ exists}\} \\
& \cup \{(v_0, w_1) : (v, w) \in E, v \in S_j \text{ or } w \in T_j\} \\
& \cup \{(\sigma, \tau) : j = 1\},
\end{aligned}
\qquad j = 0, 1.$$

We have $f = 1$ and the size of the desired time-step matching is $\ell = 2|V - ST| + |T_2| + 2$.

The rest of the argument follows (e) and so is omitted. \square

A.5 Finding an optimal set of packets is polynomial for fixed N

This appendix proves Theorem 6.3

Proof: Over a future horizon of the next f time slots, a single queue, q_{io} , lies between two extremes. At one extreme it does not send any packets and it has $q_{io}^{max} = q_{io} + \sum_{t=0}^{f-1} a_{io}^t$. At the other extreme, it sends a packet in every time slot in which case $q_{io}^{min} = q_{io}^{max} - (f + 1)$. Therefore for the entire switch, q^t can be one of at most $(f + 2)^{N^2}$ states. The recursion in (5) requires at most one computation of g per state per recursion step. At each step, for each state, the algorithm computes each of the $N!$ possible sends and observes the resulting next state. At step t , $h(q, A, S, t)$ and s^t is stored at each possible queue state. If multiple sends lead to the same next state, then the transition with the largest $h(q, A, S, t)$ is stored. At step $t = f + 1$, the S that maximizes J is found from the state with the maximum h .

Each transition is at most $N!(f + 2)^{N^2} O(1)$ calculations and there are $f + 1$ stages. Thus for a fixed N , the algorithm is $O(f^{N^2+1})$. \square

A.6 The 2-input packet arbitration problem has an $O(T)$ solution.

This appendix proves Theorem 6.4.

Proof: This constructive proof gives an algorithm based on Tables 2 and 3. Create two FIFO queues labeled C_1 and C_2 , where queue C_o contains the time when o -choices are generated in the choice state. Set the choice state to $(0, 0)$. Starting with a^1 , process the arrivals in order, one at a time. At a^t , use the tables either to immediately assign a permutation to p^t

or to generate a new choice and update the choice state. In the latter case, p^t is temporarily left unassigned. If an o -choice is generated, t is added to C_o . If an o -choice is required, then t' is taken from the front of C_o and a p^t is temporarily assigned a pointer to t' . If the choice at t' is itself a pointer to an earlier choice, then a linked list of pending assignments is created that can not be resolved until later. When p^t is assigned, if a pointer to t' is at t , then $p^{t'} = \overline{p^t}$. If a^T is processed and the choice state is (c_1, c_2) , pad the arrival sequence with $\max\{c_1, c_2\}$ empty arrivals and finish resolving the remaining choices. Since at time T , $c_o \leq T$, at most $2T$ total arrivals need to be processed.

The algorithm storage is the $O(T)$ size of C_1, C_2, P , and the pointers to earlier choices. Every a^t requires $O(1)$ processing to determine p^t immediately, to add or remove from C_o , and to set pointers to earlier choices. Chains of pointers resolve multiple p^t in a single step, but every p^t is only set once. So processing all a^t and setting every p^t requires $O(T)$ time.

□

Note that when a_t is a Case 5 or 6 arrival in Table 2, the resolution of p^t can be delayed arbitrarily far into the future by inserting Case 2 arrivals after a_t .

A.7 Maximum sequences are optimal for $N = 2$

This appendix proves Theorem 7.1.

Proof: A *forced wait* is when a packet that could be added to s (without violating constraints) is not sent. Forced waits are not allowed since they only increase the mean wait time. Hence, they can never yield an optimal sequence. In particular, at least one packet is always sent when packets are waiting.

Suppose a counterexample, (q, A, S) , to the theorem exist. Let t be the first time slot where $|s^t| < \mu(q_t)$. $|s^t| = 1$ because $|s^t| = 0$ would be a forced wait and $|s^t| = 2$ would be the maximum possible. Only two permutations are possible with a 2-input switch. The chosen permutation, p , only sends 1 packet, while the other permutation \overline{p} sends 2 packets (but, was not chosen in the non-maximum sequence S). Starting at time t , S consists of $i \geq 1$ steps of sending using p before sending using \overline{p} . The first step only sends $|s^t| = 1$ packet. Since

the packets sent by each permutation are disjoint, it cannot decrease the number of packets sent nor can it increase the average wait time by sending the 2 packets with \bar{p} permutation before the i steps using p . In this way, we can redefine S such that $|s^t| = 2 = \mu(q_t)$ without decreasing the performance criteria. By continuation, we can redefine S to be a maximum sequence without decreasing performance so that S cannot be a counterexample. \square

Acknowledgments:

This paper was motivated by preliminary studies assisted by Andrea Nagy. Some simulation studies were assisted by Narendra Ravula. Matthew Lovell provided background on optical switching. The authors would like to thank the referees for their thorough reviews.

References

- [1] Adas, A.M., "Using adaptive linear prediction to support real-time VBR video under RCBR network service model," *IEEE-ACM T. on Netw.*, v. 6, n. 5, 635–44 Oct. 1998.
- [2] Boel, R.K., James, M.R., Mareels, I.M.Y., "Performance comparison of adaptive and robust predictors for long range dependent signals," in *Proc. of the 39th IEEE Conf. on Decision and Control*, Sydney, Dec. 2000, v. 1, pp. 73–8.
- [3] Brown, T.X, Lui, K.H., "Neural Network Design of a Banyan Network Controller," *IEEE J. on Selected Areas in Comm.*, v. 8, n. 8, pp. 1428–38, Oct., 1990.
- [4] Brown, T.X, "Switch Packet Arbitration via Queue Learning," *Advances in Neural Information Processing Systems 14*, ed. T.G. Dietterich et al., MIT Press, 2002.
- [5] Chuang, S.-T., Goel, A., McKeown, N., Prabhakar, B., "Matching Output Queueing with a Combined Input and Output Queued Switch," in *Proc. IEEE INFOCOM*, 1999.
- [6] Endo, N., Kozaki, T., Ohuchi, T., Kuwahara, H., Gohara S., "Shared Buffer Memory Switch For An ATM Exchange," *IEEE T. on Comm.*, v. 41, pp. 237–45, Jan. 1993.
- [7] Even, S., Itai, A., and Shamir, A., "On the complexity of timetable and multicommodity flow problems," *SIAM J. on Computing*, v. 5, n. 4, 691–703, Dec., 1976.
- [8] Fan, Z., Mars, P., "Access flow control scheme for ATM networks using neural-network-based traffic prediction," *IEE Proc.-Comm.*, v. 144, n. 5, 295–300, Oct. 1997.
- [9] Garrett, M.W., Willinger, W., "Analysis, Modeling, and Generation of Self-Similar VBR Video Traffic," in *Proc. of ACM SIGCOMM*, 1994, pp. 269–80.
- [10] Giacomelli, J. N., Sincoskie, W. D., Littlewood, M., "Sunshine: A High Performance Self-Routing Broadband Packet Switch Architecture," *International Switching Symposium*, Paper 138, pp. 1–6, Jun. 1990.
- [11] Hluchyj, M. G., Karol, M. J., "Queueing in high-performance packet switching," *IEEE J. Selected Areas in Comm.*, v. 6, n. 9 pp. 1587–97, Dec. 1988.

- [12] Hopcroft, J. and Karp, R., “An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs”, *SIAM J. Computing* v. 2, n. 4, 1973, pp 225–31.
- [13] Karol, M. J., Hluchyj, M. G., Morgan, S. P., “Input vs. Output Queueing on a Space-Division packet Switch,” *IEEE T. on Comm.*, v. 35, pp. 1347–56, Dec. 1987.
- [14] Leland, W.E., Taqqu, M.S., Willinger, W., Wilson, D.V., “On the Self-Similar Nature of Ethernet Traffic,” in *Proc. of ACM SIGCOMM*, 1993, pp. 183–93.
- [15] McKeown, N., Anantharam, V., Walrand, J., “Achieving 100% Throughput in an Input-Queued Switch,” in *Proc. IEEE INFOCOMM*, San Francisco, Mar. 1996.
- [16] McKeown, N., Anderson, T.E., “A quantitative comparison of iterative scheduling algorithms for input-queued switches,” *Computer Networks and ISDN Systems*, v. 30, n. 24, Dec. 14, 1998, pp. 2309–26.
- [17] McKeown, N., Prabhakar, B., Zhu, M., “Matching Output Queueing with Combined Input and Output Queueing.” in *Proc. 35th Annual Allerton Conf. on Communications, Control, and Computing*, Monticello, IL, Oct. 1997.
- [18] Oie, Y., Murata, M., Kubota, K., Miyahara, H., “Effect of speedup in non-blocking packet switch,” in *Proc. ICC ‘89*, Boston, MA, June 1989, p. 410–4.
- [19] Park, Y.-K., Lee, G., “NN Based ATM Scheduling with Queue Length Based Priority Scheme,” *IEEE J. Selected Areas in Comm.*, v. 15, n. 2 pp. 261–70, Feb. 1997.
- [20] Partridge, C., Carvey, P.P., et al., “A 50Gb/s IP Router,” *IEEE/ACM T. on Networking*, v. 6, n. 3, Jun. 1998, pp. 237–48.
- [21] Prycker, M. de, *Asynchronous Transfer Mode: Solution for Broadband ISDN*, Prentice Hall, Englewood Cliffs, New Jersey, 1995.
- [22] Qiao, C., Yoo, M., “Optical Burst Switching (OBS)—A new paradigm for an optical internet,” *J. of High Speed Networks*, v. 8, n. 1, pp. 69–84, 1999.
- [23] Ramanan, A.V., Jordan, H.F., Sauer, J.R., “Space-Time Networks for Optical Communications,” *Proc. 31st Allerton Conf. on Communication, Control and Computing*, pp. 566–75, Univ. of Illinois, Urbana-Champaign, IL, Sept. 1993.
- [24] Schoenen, R., “An architecture supporting quality-of-service in virtual-output-queued switches,” *IEICE T. on Comm.*, v. E83-B, n. 2, Feb. 2000, pp. 171–81.
- [25] Tamir, Y., Chi, H.-S., “Symmetric crossbar arbiters for VLSI communication switches,” *IEEE T. on Parallel and Dist. Sys.*, v. 4, n. 1, Jan. 1993, pp. 13–27.
- [26] Tassiulas, L., “Linear complexity algorithms for maximum throughput in radio networks and input queued switches,” in *Proc. IEEE INFOCOM*, 1998, pp. 533–9
- [27] Troudet, T.P., Walters, S.M., “Neural Network Architecture for Crossbar Switch Control,” *IEEE T. on Circuits and Systems*, v. 38, n. 1, Jan. 1991, pp. 42–56.
- [28] Viswanathan, A., Feldman, N., Wang, Zheng, Callon, R., “Evolution of multi-protocol label switching,” *IEEE Comm. Mag.*, v. 36, n. 5, pp. 165–73, May 1998.
- [29] Yoo, M., Qiao, C., “QoS performance of optical burst switching in IP-over-WDM networks,” *IEEE J. on Selected Areas in Comm.*, v. 18, n. 10, pp. 2062–71, Oct. 2000.