Pipelined Two Step Iterative Matching Algorithms for CIOQ Crossbar Switches *

Deng Pan
Dept. of Computer Science
State University of New York
Stony Brook, NY 11794, USA
pandeng@cs.sunysb.edu

Yuanyuan Yang
Dept. of Electrical & Computer Engineering
State University of New York
Stony Brook, NY 11794, USA
yang@ece.sunysb.edu

ABSTRACT

Traditional iterative matching algorithms for VOQ switches need three steps, i.e., request, grant and accept. By incorporating arbitration into the request step, two step iterative matching can be achieved. This enables simpler implementation and shorter scheduling time, while maintaining almost identical performance. As an example of the two step iterative matching algorithms, in this paper we present Two Step Parallel Iterative Matching (PIM2), and theoretically prove that its average convergence iterations are less than $\ln N + e/(e-1)$ for an $N \times N$ switch. Furthermore, two step iterative matching algorithms can be efficiently pipelined on CIOQ switches so that two matchings can be obtained in each time slot. We propose a scheme called Second of Line (SOL) matching to provide two independent virtual switches, with which the pipelining can be achieved without additional scheduling time and arbitration hardware. More importantly, the pipelined algorithms are theoretically guaranteed to achieve 100% throughput for any admissible traffic. Extensive simulations are conducted to show that our analytical result on the average convergence iterations $\ln N + e/(e-1)$ is more accurate than the classical result $\log_2 N + 4/3$, and to test the performance of different pipelined algorithms on CIOQ switches.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Packet-switching networks

General Terms

Algorithms, Design

Keywords

Scheduling, Pipeline, Iterative algorithms, Convergence

*This research was supported in part by the U.S. National Science Foundation under grant numbers CCR-0073085, CCR-0207999 and ECS-0427345.

ANCS'05, October 26–28, 2005, Princeton, New Jersey, USA.

1. INTRODUCTION

Crossbar switches operating with fixed length packets have demonstrated advantages in high speed switching [1, 3, 7, 5, 8]. On the one hand, the crossbar fabric provides non-blocking switching capability, which is desired by high speed switches. On the other hand, fixed length packets (or cells) enable the switch to work in a synchronous time slot mode, which considerably simplifies and accelerates the scheduling and switching processes. In each time slot, the scheduling algorithm makes a scheduling decision in the form of conflict-free input-output pairs, and the crossbar simultaneously transmits all the scheduled packets.

The general structure of an $N \times N$ crossbar switch is illustrated in Figure 1. The input ports and output ports are connected by a crossbar switching fabric, which may have speedup capability. A crossbar with speedup of S can remove S packets from each input port and deliver S packets to each output port in a single time slot. Depending on the exact speedup factor, temporarily blocked packets may be buffered at either the input side or the output side or both. When $S = 1$, each output port will receive at most one packet per time slot, which can be immediately sent to the outline, and therefore no buffer space is needed at the output side. Such a switch is called an input queued (IQ) switch. When $S = N$, the crossbar is able to deliver N packets to an output port per time slot. Thus, even each input port simultaneously has a new incoming packet destined to the same output port, all of these packets can be transmitted in the same time slot. As a result, the buffer space at the input side can be eliminated, and such a switch is called an output queued (OQ) switch. When $1 < S < N$, both input ports and output ports need to have buffers, and such a switch is a combined input output queued (CIOQ) switch.

For the buffer space at the input side, we consider only the virtual output queued (VOQ) buffering [2], since the traditional single FIFO queue suffers from the head of line (HOL) blocking, i.e., even though the destination output ports of the packets behind the HOL packet may be free, they cannot be scheduled to transmit because the HOL packet is blocked. It was proved in [4] that the HOL blocking limits the switch's maximum throughput to about $2 - \sqrt{2} \approx 58.6\%$. On the contrary, the VOQ buffering maintains a (logically) separate queue for each output port at each input port, so that a packet will no longer be held up by another packet ahead of it that goes to a different output port. Conventionally, an IQ switch with VOQ buffering is called a VOQ switch.

The scheduling problem on crossbar packet switches can be viewed as a special case of the bipartite graph matching

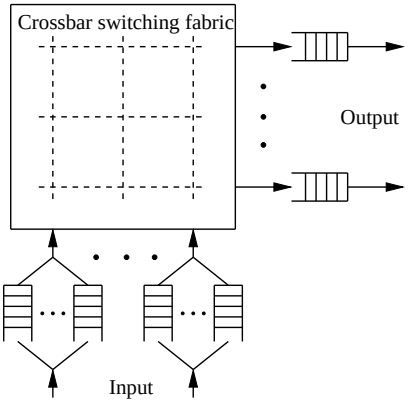


Figure 1: The general structure of a crossbar switch.

problem. Input ports and output ports make up of the two disjoint sets of vertices, and the scheduling decisions are represented by the edges from input ports to output ports. Traditionally, maximum size matching (MSM) [2] and maximum weight matching (MWM) [2] are adopted in order to maximize the throughput of the switch. However, while MWM is able to achieve 100% throughput for any independent traffic, MSM may lead to instability and unfairness under admissible traffic, and starvation under inadmissible traffic [9]. Besides, both MSM and MWM have high time complexity, which is $O(N^{2.5})$ [10] and $O(N^3 \log N)$ [11], respectively, and therefore are impractical for high speed switching.

In order to make fast scheduling decisions, iterative matching algorithms, such as PIM [3] and iSLIP [1] were proposed. The algorithms attempt to quickly converge on a maximal matching in multiple iterations. As shown in Figure 2, each iteration of the algorithms usually consists of the following three steps:

Request step. Each input port sends a packet to every output port for which it has a buffered packet.

Grant step. An output port selects one request among all the requests that it receives to grant.

Accept step. An input port selects one grant among all the grants that it receives to accept. Then, the input port marks itself and the corresponding output port as matched.

All input ports and output ports are initially unmatched and only those not matched at the end of one iteration are considered in the next round. Iterative matching algorithms find a maximal matching each time slot by incrementally adding input-output pairs, without removing the ones made earlier. In general, a maximal matching is smaller than a maximum matching (the one with largest size or weight), but is easier to obtain. Also, it was proved in [6] that, for a CIOQ switch with speed up of two, any maximal matching achieves 100% throughput.

The purpose of this paper is to design efficient iterative matching algorithms for CIOQ crossbar switches. First, we show that by incorporating arbitration into the request step, two step iterative matching can be achieved, as shown in Figure 3, which enables simpler implementation and shorter scheduling time, while maintaining almost the same performance as three step algorithms. As an example, we present Two Step Parallel Iterative Matching (PIM2) for VOQ switches, which is the two step version of PIM [3], and prove that its average convergence iterations are less than $\ln N + e/(e-1)$. In order to efficiently apply two step iterative matching algorithms to CIOQ switches, we propose a scheme called Second of Line (SOL) matching which makes two scheduling decisions in one time slot, without extra scheduling time or arbitration hardware requirement. We prove that any two step iterative matching algorithm pipelined with the proposed scheme achieves 100% throughput for any admissible traffic. In order to further reduce the packet transmission delay, a mechanism called HOL rescheduling is also proposed. Extensive simulations are conducted to verify the accuracy of our analysis on the convergence property of PIM2, and to compare the performance of different scheduling algorithms on CIOQ switches.

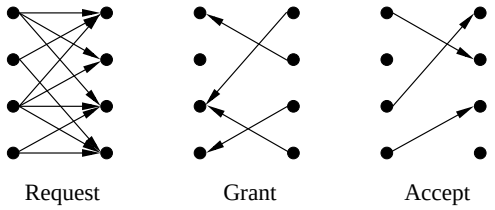


Figure 2: Three step iterative matching.

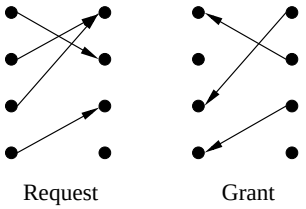


Figure 3: Two step iterative matching.

The rest of this paper is organized as follows. Section 2 discusses the two step iterative matching algorithms for VOQ switches. Section 3 applies the two step algorithms to CIOQ switches in a pipelined manner. Section 4 gives the simulation results. Finally, Section 5 concludes the paper.

2. TWO STEP ITERATIVE MATCHING FOR VOQ SWITCHES

In this section, we first analyze the advantages of two step iterative matching algorithms. Then, as an example, we present Two Step Parallel Iterative Matching (PIM2) for VOQ switches, and theoretically prove that its average convergence iterations are less than $\ln N + e/(e-1)$. Other generalizations of the two step iterative matching algorithms are discussed as well in this section.

2.1 Advantages of Two Step Iterative Matching

For the traditional three step iterative matching, each input port can send up to N requests and receive up to N grants. Thus, the accept step is necessary for each input port to choose one grant among the possible N grants. Alternatively, if the arbitration in the accept step is executed before even sending out requests, so that each input port sends only one request and correspondingly receives only one grant, the accept step can be eliminated.

Comparing with the traditional three step algorithms, the two step iterative matching has some advantages which we summarize as follows. Firstly, by eliminating the accept step, the time for one iteration of the algorithm is reduced, and thus shorter total scheduling time can be achieved. Secondly, for a two step iterative matching algorithm, the request step and the accept step are carried out the same way, i.e., to

arbitrate among N items and choose one. This property enables easier and cheaper implementation of the two step algorithms. Thirdly, since each input port sends only one request, the data exchanged between input ports and output ports are greatly reduced. Especially, since each input port can get at most one grant, the request step of the next iteration can start immediately after the only grant is received. While for the three step algorithms, an output port needs to wait for up to N requests before begins the grant step, and an input port needs to wait for up to N grants before begins the accept step.

2.2 PIM2

As an example, we present a two step iterative matching algorithm called Two Step Parallel Iterative Matching (PIM2) for VOQ switches, which corresponds to the three step algorithm PIM in [3]. Each iteration of the PIM2 algorithm includes the following two steps:

Request step. Each input port randomly sends a request to an output port for which it has a buffered packet.

Grant step. An output port randomly grants to one request among all requests it receives. The output port marks itself and the corresponding input port as matched.

Similarly, all input ports and output ports are initially unmatched and only those not matched at the beginning of an iteration will be considered. The algorithm continues until there is no more matchable input-output pairs. Thus, a maximal matching has been found.

As in PIM, the request or grant arbitrations of different input ports or output ports are independent, and therefore can be done in parallel to accelerate the matching process. Also, PIM2 makes arbitration decisions on a random basis, so each input port has equal transmission opportunity, and fairness is achieved.

2.3 Convergence Property of PIM2

Because of the simplicity, the PIM2 algorithm is easy to analyze. In this subsection, we discuss its convergence property.

One perception to a two step iterative matching algorithm might be that since an input port sends much less requests in each iteration, the algorithm may take longer time to converge, which is of course unfavorable for a practical scheduling algorithm. However, our following theoretical analysis and simulations in Section 4 both show that three step algorithms and two step algorithms have almost identical convergence properties. To understand this, we can view the three step iterative matching as that in the grant step each output port selects one input port to "request," and in the accept step, each input port selects one output port to "grant." Then, three step iterative matching is only different from two step iterative matching in its extra request step.

In the following analysis, we assume a uniformly distributed traffic model. We first define some notations to represent the matching status. An input port is said to be free if it is not matched, but has buffered packets to an unmatched output port, and similarly, an output port is free if it is not matched, but at least one free input has packets to it.

in_i: the i^{th} input port;
out_j: the j^{th} output port;
q_{ij}: the queue of in_i that buffers packets to out_j;
$fanout(in_i) = \{out_j | out_j$ is free, and q_{ij} is not empty$\}$;
$FreeIn(k) = \{in_i | in_i$ is free after k iterations$\}$;
$O(FreeIn(k)) = \bigcup_{in_i \in FreeIn(k)} fanout(in_i)$;
$p(k) = \min\{|FreeIn(k)|, |O(FreeIn(k))|\}$.

$p(k)$ is the largest possible number of input-output pairs that still can be matched in the current time slot after k iterations, and we call it potential.

LEMMA 1. *After one more iteration, the expected value of the new potential is $1/e$ of that before this iteration, i.e.,*

$$E(p(k+1)) \le \frac{p(k)}{e}$$

Proof. Suppose that after k iterations, $|FreeIn(k)| = m$ and $|O(FreeIn(k))| = n$. And assume that, for the average case, the fanout of a free input port is uniformly distributed among the rest of the free output ports. Thus, the probability for a free output out_j to receive the request from a free input in_i in the $(k+1)^{th}$ iteration is

$$\begin{aligned} & Pr\{out_j \text{ receives a request from } in_i\} \\ = & \frac{|fanout(in_i)|}{n} \times \frac{1}{|fanout(in_i)|} \\ = & \frac{1}{n} \end{aligned}$$

The first part $\frac{|fanout(in_i)|}{n}$ of the formula is the probability that out_j is in the fanout of in_i, and the second part $\frac{1}{|fanout(in_i)|}$ is the probability that in_i sends the request to any output port of its fanout. Thus, the probability that a free output does not receive any request in the $(k+1)^{th}$ iteration is

$$Pr\{out_j \text{ does not receive any request}\} = \left(1 - \frac{1}{n}\right)^m$$

and we obtain

$$\begin{aligned} & E(|O(FreeIn(k+1))|) \\ = & |O(FreeIn(k))| \times Pr\{\text{an output receives no request}\} \\ = & n \times \left(1 - \frac{1}{n}\right)^m \end{aligned}$$

In order words, the expected value of the number of free output ports after the $(k+1)^{th}$ iteration is $n\left(1-\frac{1}{n}\right)^m$.

According to the definition of the potential, $p(k)$ is equal to the smaller of m and n. In the following, we will discuss two possible cases.

Case 1: $m \ge n$ and $p(k) = n$.

Define $f(n) = \left(1-\frac{1}{n}\right)^n$. It has a limit when n goes to infinity:

$$\lim_{n\to\infty} f(n) = \lim_{n\to\infty} \left(1 - \frac{1}{n}\right)^n = \frac{1}{e}$$

and it is easy to verify that for any practical n, say, $n \le 10^6$, $f(n) \le \frac{1}{e}$. Thus, we can obtain

$$\begin{aligned} & E(p(k+1)) \\ \le & E(|O(FreeIn(k+1))|) = n\left(1 - \frac{1}{n}\right)^m \\ \le & n\left(1 - \frac{1}{n}\right)^n \le \frac{n}{e} = \frac{p(k)}{e} \end{aligned}$$

Case 2: $m < n$ and $p(k) = m$.

For $m = 1$, it is trivial that the algorithm converges after one more iteration. We consider in the following $m \ge 2$.

Since the expected value of the number of input-output pairs matched in the $(k+1)^{th}$ iteration is $n - n(1-\frac{1}{n})^m$, we have $E(FreeIn(k+1)) = m - n + n(1-\frac{1}{n})^m$.

Define $g(x) = m - x + x(1-\frac{1}{x})^m - m/e$, and $g'(x) = -1 + (1-\frac{1}{x})^m + (1-\frac{1}{x})^{m-1}\frac{m}{x}$. Define $h(m) = (1-\frac{1}{x})^{m-1}\frac{x+m-1}{x}$. When $m = 2$, $h(2) = 1 - \frac{1}{x^2} < 1$ for any $x \ne 0$; when

$m > 2$, it is easy to prove by induction that $h(m) < h(2) < 1$. Thus, we have $g'(x) < 0$ for any $m \geq 2$. Since $g(m) = m(1 - \frac{1}{m})^m - \frac{m}{e} < 0$, we have $g(n) < g(m) < 0$. In other words,

$$m - n + n\left(1 - \frac{1}{n}\right)^m \leq \frac{m}{e}$$

Therefore, we can obtain

$$\begin{aligned} & E(p(k+1)) \\ \leq\ & E(O(FreeIn(k+1))) = m - n + n\left(1 - \frac{1}{n}\right)^m \\ \leq\ & \frac{m}{e} = \frac{p(k)}{e} \end{aligned}$$

Thus, in each case, we have $E(p(k+1)) \leq p(k)/e$. ■

LEMMA 2. *Assume $M \geq N$. Then for an $N \times M$ or $M \times N$ switch, the expected value of the potential after k iterations is less than or equal to N/e^k, i.e.,*

$$E(p(k)) \leq \frac{N}{e^k}$$

Proof. We prove it by induction.

Base case: When $i = 0$, i.e., before any matching has been done, we have $E(p(0)) \leq N$.

Inductive case: Suppose $E(p(k)) \leq N/e^k$ holds. From Lemma 1, we know that $E(p(k+1)|p(k)) \leq p(k)/e$. Then,

$$\begin{aligned} & E(p(k+1)) = E(E(p(k+1)|p(k))) \\ \leq\ & E(\frac{p(k)}{e}) = \frac{E(p(k))}{e} \end{aligned}$$

By the inductive hypothesis, $E(p(k+1)) \leq \frac{N}{e^{k+1}}$. ■

Define C to be the number of convergence iterations of PIM2. We have the following theorem regarding the average value of C.

THEOREM 1. *Assume $M \geq N$. Then for an $N \times M$ or $M \times N$ switch, the average number of convergence iterations of PIM2 is less than or equal to $\ln N + e/(e-1)$.*

Proof. Since the potential is decreased by at least one in each iteration, it is clear that C is in the range $[1, N]$. Therefore

$$\begin{aligned} E(C) &= \sum_{i=1}^{N} i \times Pr\{C = i\} = \sum_{j=1}^{N}\sum_{i=j}^{N} Pr\{C = i\} \\ &= \sum_{j=1}^{N} Pr\{j \leq C \leq N\} \end{aligned}$$

On the other hand,

$$\begin{aligned} Pr\{j \leq C \leq N\} &= \sum_{k=1}^{N-j+1} Pr\{p(j-1) = k\} \\ &\leq \sum_{k=1}^{N-j+1} k \times Pr\{p(j-1) = k\} \\ &= E(p(j)) \leq \frac{N}{e^j} \end{aligned}$$

Also, because $Pr\{j \leq C \leq N\} \leq 1$, we have

$$E(C) \leq \sum_{j=1}^{N} \min\left\{1, \frac{N}{e^j}\right\} \leq \ln N + \frac{e}{e-1}$$

■

2.4 Generalization of Two Step Iterative Matching

The basic idea of two step iterative matching can be generalized to other existing three step algorithms as well. For example, the well known iSLIP algorithm [1] improves upon PIM [3] by making arbitration based on round robin pointers, which automatically adapt to different input ports or output ports under heavy load so that fast scheduling decisions can be made. As an example, we give the two step version of iSLIP, which we call iSLIP2, as follows.

Request step. Each free input port sends a request to the first free output port which appears next to its round robin pointer and it has buffered packets destined to.

Grant step. Each free output port chooses the request from the first input port which appears next to its round robin pointer, and grants it to transmit. For the first iteration of each time slot, the round robin pointers of the newly matched input port and output port are both incremented by one (in a modular manner).

Similarly, under heavy load, the round robin pointers of different input ports or output ports in iSLIP2 also tend to desynchronize with respect to one another. More importantly, iSLIP2 does not have any extra "overhead" in this scenario. In other words, all the N requests are granted, while in iSLIP N^2 requests are sent, but only N of them are granted and the rest of $N(N-1)$ are unnecessary overhead. Thus, iSLIP2 is more efficient in this sense.

The two step iterative matching can also be used to schedule multicast traffic [13]. The crossbar switch can have built-in capability to simultaneously send a packet from one input port to multiple output ports to efficiently support multicast communication. In order to apply two step iterative matching to multicast scheduling, each input port sends requests to all the destination output ports of its earliest packet, and each output port also grants to the packet with the smallest arrival time. Hence, the chance of the earliest multicast packet in the switch being delivered to all its output ports in the same time slot is increased. Besides, because all the requests sent by an input port are for the same multicast packet, there is no potential transmission conflict. As indicated in [13], the two step multicast iterative matching algorithm has small average convergence iterations and achieves short multicast latency as well.

For CIOQ switches, scheduling algorithms are required to run faster in order to make more than one scheduling decisions each time slot within the same scheduling time limitation as that under VOQ switches. Two step iterative matching algorithms can also be efficiently adapted to schedule packets for CIOQ switches, which we will discuss in detail in the next section.

3. PIPELINED TWO STEP ITERATIVE MATCHING FOR CIOQ SWITCHES

In this section, we propose a scheduling scheme, called Second of Line (SOL) matching, for CIOQ switches with speedup of two. Different from any specific scheduling algorithm, such as [12], our scheme can efficiently pipeline any two step iterative matching algorithm, so that two scheduling decisions can be made in each time slot without extra scheduling time or arbitration hardware requirement.

On VOQ switches, the request step and grant step of a two step iterative matching algorithm can only be executed in a sequential manner, as shown in Figure 4. The reason is

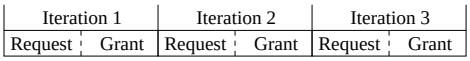


Figure 4: On VOQ switches, the request step and the grant step have to progress in a sequential manner.

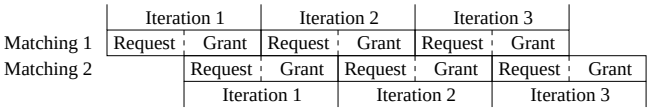


Figure 5: On CIOQ switches, the request step and the grant step of different matching processes can be pipelined to fully utilize the arbitration logic.

that only free input ports can send requests in the request step, but whether an input port is free or not is not known until the grant step of the previous iteration is completed. Therefore, the request logic and grant logic are only busy for a half of the total time and are not fully utilized.

For CIOQ switches with speedup of two, the scheduling algorithm needs to obtain two matchings in each single time slot. If the two matchings are arranged in such a way that the request step of one is coincident with the grant step of the other, and vice versa, as illustrated in Figure 5, the arbitration logic can be fully pipelined and two matchings can be obtained within the same scheduling time as in the sequential case plus an extra step time. As we will see in the simulation section (Section 4), because of the speedup, the pipelined algorithms usually have a smaller number of convergence iterations, and the two matchings can be obtained within the same scheduling time limitation as that under VOQ switches.

3.1 HOL Matching and SOL Matching

Because the two matching processes progress simultaneously, two "virtual" switches are needed so that each matching can independently work on one of the virtual switches. This is done by adding lookahead information into each virtual queue. In addition to the head of line (HOL) packet of each virtual queue, the second of line (SOL) packet is checked as well. For easy understanding, we can view the two matchings as being made for the HOL packets and the SOL packets respectively, as shown in Figure 6. Since the HOL matching and SOL matching are completely independent, each of them does not need to wait for the information from the other, and they can run in parallel, as long as the logic needed is idle. To be precise, the HOL matching starts first. While the HOL matching progresses to the grant step, as we discussed above, the request logic is idle. Then, the request step of the SOL matching can start. When the grant step of the HOL matching is completed, the SOL matching has also finished its request step. Thus, they can simultaneously move to the next steps, and so on. This pipelined process continues until both the HOL matching and the SOL matching converge.

It should be noted that although the SOL matching is based on the SOL packet information, the actually transmitted packets with the SOL matching result are not necessarily the SOL packets. After the HOL matching is finished, the crossbar extracts the first packets of all the scheduled virtual queues and simultaneously transmits them to the scheduled output ports. After one or more steps, the SOL matching is done as well, and again, the crossbar simply removes the first packet from each corresponding virtual queue. The transmission will not introduce any conflict, since the scheduling algorithms make decisions based on the number of packets in each virtual queue and do not care about the exact content of the packets. In this way, the packets of the same virtual queue are transmitted in their arrival order, and out-of-sequence delivery is avoided.

3.2 Throughput of Pipelined Two Step Iterative Matching Algorithms

By using the fluid model theory in [6], it can be shown that any two step iterative matching algorithm pipelined with the above proposed scheme achieves 100% throughput for any admissible traffic.

Use $A_{ij}(n)$ and $D_{ij}(n)$ to denote the number of packet arrivals and departures at q_{ij} up to time slot n, respectively. Conventionally, $A_{ij}(0) = 0$ and $D_{ij}(0) = 0$. $A_{ij}(n)$ satisfies a strong law of large numbers, i.e.,

$$\lim_{n\to\infty} \frac{A_{ij}(n)}{n} = \lambda_{ij}$$

where λ_{ij} is called the arrival rate of q_{ij}. A traffic is said to be admissible if it has no oversubscription at any input port or output port, i.e.,

$$\forall i, \sum_{j'=1}^{N} \lambda_{ij'} \le 1, \text{ and } \forall j, \sum_{i'=1}^{N} \lambda_{i'j} \le 1 \qquad (1)$$

Define $Z_{ij}(n) = A_{ij}(n) - D_{ij}(n)$, which is the number of packets in q_{ij} at time slot n, and $C_{ij}(n) = \sum_{j'=1}^{N} Z_{ij'}(n) + \sum_{i'=1}^{N} Z_{i'j}(n)$, which is the total number of packets in the virtual queues of input port in_i and in the virtual queues to output port out_j of different input ports. Based on the fluid model in [6], we have the following lemma.

LEMMA 3. *For any pipelined two step iterative matching algorithm on a CIOQ switch with speedup of two, the following fluid equation holds*

$$\dot{C}_{ij}(t) \le \sum_{j'=1}^{N} \lambda_{ij'} + \sum_{i'=1}^{N} \lambda_{i'j} - 2$$

whenever $Z_{ij}(t) > 0$.

Proof. By the fluid limit procedure, it is equivalent to show that, if $Z_{ij}(n) \ge 2$,

$$C_{ij}(n+1) - C_{ij}(n) \le \sum_{j'=1}^{N} (A_{ij'}(n+1) - A_{ij'}(n)) + \sum_{i'=1}^{N} (A_{i'j}(n+1) - A_{i'j}(n)) - 2$$

Since $Z_{ij}(n) \ge 2$, there must be a HOL packet as well as a SOL packet at q_{ij}. Therefore, at least one packet should be scheduled to transmit either from in_i or to out_j in both the HOL matching and the SOL matching. Otherwise, both in_i and out_j are free, but the HOL (SOL) packet of q_{ij} is not scheduled in the HOL (SOL) matching, which contradicts the matching process. Use $\pi_{ij}(n)$ and $\pi'_{ij}(n)$ to denote the HOL and SOL matching results for q_{ij} at time slot n, respectively. For example, if $\pi_{ij}(n) = 1$, the HOL packet of q_{ij} is scheduled to be sent from in_i to out_j at slot n in the HOL matching. Then, according to the above reasoning, we can obtain

$$\sum_{j'=1}^{N} \pi_{ij'}(n) + \sum_{i'=1}^{N} \pi_{i'j}(n) \ge 1, \text{ and } \sum_{j'=1}^{N} \pi'_{ij'}(n) + \sum_{i'=1}^{N} \pi'_{i'j}(n) \ge 1$$

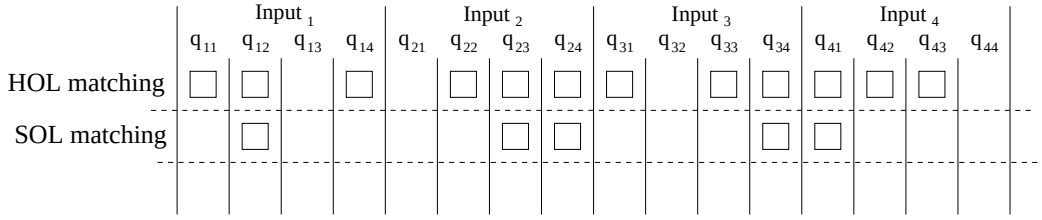


Figure 6: By adding lookahead information, two matchings can independently work on the HOL packets and SOL packets respectively.

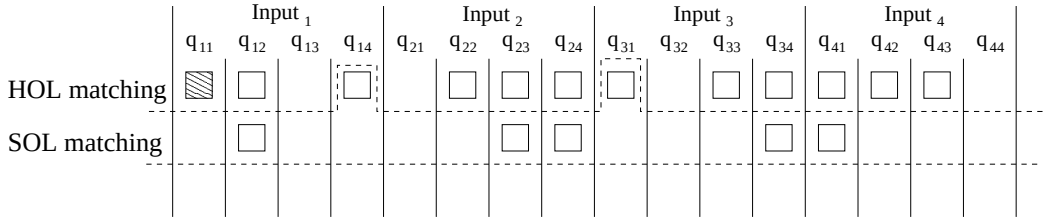


Figure 7: By allowing the HOL packets that lose competition in the HOL matching to participate in the SOL matching, the packet delay can be reduced and the throughput of the SOL matching can be improved.

Since for a pipelined algorithm, $D_{ij}(n+1)-D_{ij}(n) = \pi_{ij}(n)+\pi'_{ij}(n)$, we have

$$\sum_{j'=1}^{N}(D_{ij'}(n+1)-D_{ij'}(n))+\sum_{i'=1}^{N}(D_{i'j}(n+1)-D_{i'j}(n)) \geq 2$$

Finally, by the definition of $C_{ij}(n)$, it follows that

$$\begin{aligned}
& C_{ij}(n+1)-C_{ij}(n) \\
= & \sum_{j'=1}^{N}(A_{ij'}(n+1)-D_{ij'}(n+1)) \\
+ & \sum_{i'=1}^{N}(A_{i'j}(n+1)-D_{i'j}(n+1)) \\
- & \sum_{j'=1}^{N}(A_{ij'}(n)-D_{ij'}(n))-\sum_{i'=1}^{N}(A_{i'j}(n)-D_{i'j}(n)) \\
\leq & \sum_{j'=1}^{N}(A_{ij'}(n+1)-A_{ij'}(n))+\sum_{i'=1}^{N}(A_{i'j}(n+1)-A_{i'j}(n))-2
\end{aligned}$$

■

THEOREM 2. *Any pipelined two step iterative matching algorithm on a CIOQ switch with speedup of two achieves 100% throughput for any admissible traffic.*

Proof. By Lemma 3 and the no oversubscription condition (1), we have

$$\dot{C}_{ij}(t) \leq \sum_{j'=1}^{N}\lambda_{ij'}+\sum_{i'=1}^{N}\lambda_{i'j}-2 \leq 0$$

Following the proof of Theorem 2 in [6], it can be shown that $Z_{ij}(t) = 0$ for almost every $t \geq 0$, which means that the switch is stable. The rest of the proof is omitted in this paper. ■

3.3 HOL Rescheduling

By rescheduling the failed HOL packets in the SOL matching, the packet delay can be further reduced. Consider an extreme case, where any virtual queue has at most one packet, i.e., the HOL packet. In this case, the SOL matching does not generate any result, because there is no SOL packet. However, for the HOL matching, at most one HOL packet will be matched among all the virtual queues of the same input port or those to the same output port. For the HOL packets that fail in the competition for the HOL matching, they can still participate in the subsequent SOL matching of the same time slot to improve the efficiency of the algorithm. We call the mechanism "HOL rescheduling," i.e., to allow failed HOL packets to be rescheduled in the following SOL matching iterations. An example of HOL rescheduling is shown in Figure 7. Originally, there are only five SOL packets. Suppose, at some time, the HOL packet of q_{11} is matched in the HOL matching. As a result, all other HOL packets at the virtual queues of in_1 and the virtual queues to out_1 have no chance to be matched in the HOL matching anymore, which we would like to reconsider in the SOL matching. For q_{12} and q_{13}, there is no change after HOL rescheduling, since q_{12} already has a SOL packet while q_{13} does not have a HOL packet. But for q_{14}, there is no SOL packet originally, and therefore the failed HOL packet can improve the possible matching pairs of the SOL matching. Similarly, the HOL packet of q_{31} can participate in the SOL matching while there is no change to q_{21} and q_{41}. The side effect of HOL rescheduling is that the SOL matching may take more iterations to converge. Fortunately, as can be seen in the simulations section (Section 4), it only slightly increases the convergence iterations.

3.4 Hardware Implementation

For a practical scheduling algorithm, easy and efficient implementation is important. As illustrated in Figure 8, the proposed pipeline scheme can be efficiently implemented in hardware. Since the HOL matching and SOL matching are progressing simultaneously, two sets of registers are needed to keep the input states of the two matchings and another two sets of registers are needed to store the matching decisions. However, no extra arbitration logic is necessary, since the two matchings can be pipelined to make fully use of the single set of hardware. For PIM2 and iSLIP2, their request and grant arbitrations perform the same operation, and the arbiters can be efficiently implemented.

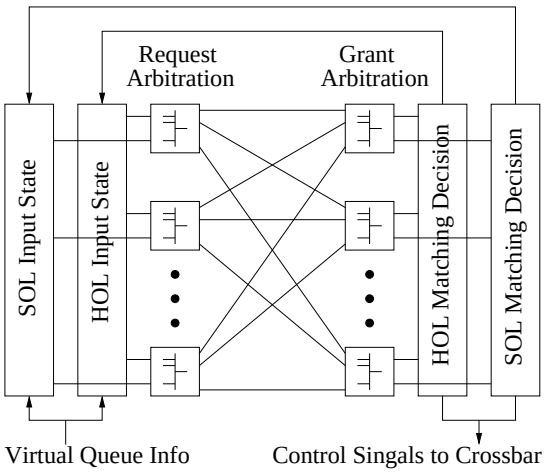


Figure 8: High level diagram of the hardware implementation of the proposed pipeline scheme.

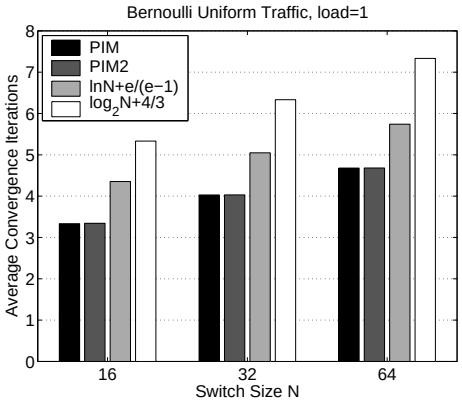
At the beginning of each time slot, the HOL and SOL input state registers are initialized according to the occupancy of the corresponding virtual queues. Then, the two sets of input state registers alternately use the request arbitration logic to send requests, and the grant arbitration logic will continuously send matched input-output pairs to the two sets of matching decision registers in a pipelined manner. The process goes on until both matchings converge. Finally, the matching decisions are forwarded to the crossbar as control signals to transmit the scheduled packets.

4. SIMULATION RESULTS

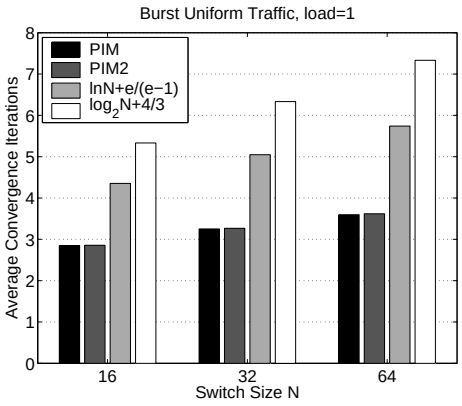
Extensive simulations are conducted to verify the accuracy of the convergence iteration analysis in Section 2, and to test the performance of the pipelined two step iterative matching algorithms.

We consider both Bernoulli arrival and burst arrival. Bernoulli arrival is one of the most widely used models in the simulation of scheduling algorithms. With Bernoulli arrival, each input port has the probability of p to have a new packet to arrive at the beginning of a time slot. However, in practice, network packets are usually highly correlated and tend to arrive in a burst mode. The burst nature can be described by a Markov process alternating between off and on states. In the off state, there is no packet to arrive. In the on state, packets arrive at every time slot and all have the same destinations. At the end of each time slot, the traffic can switch between off and on states independently. Burst traffic can be described using two parameters E_{off} and E_{on}. E_{off} is the average length of the off state, or alternatively the probability to switch from the off state to the on state is $1/E_{off}$. E_{on} is the average length of the on state, or the probability to switch from the on state to the off state is $1/E_{on}$. Therefore, the arrival rate is $E_{on}/(E_{off} + E_{on})$.

In the simulations, both uniform traffic and non-uniform traffic (hotspot traffic [14] in our case) are adopted. For uniform traffic, the destination of a new incoming packet is uniformly distributed among all the output ports, i.e., $\lambda_{ij} = p/N$, where p is the arrival rate. For hotspot traffic, each input port has a "hotspot" output port, which is the destination of a half of the arriving packets, and the rest of output ports receive equal amount of packets. In our simulations, we set $\lambda_{ii} = p/2$ and $\lambda_{ij} = p/2(N-1)$ for $i \neq j$.



(a) Bernoulli uniform traffic



(b) Burst uniform traffic

Figure 9: Comparison of average convergence iterations with different switch sizes.

Each virtual queue of the switch is set to be able to hold a maximum number of 10^4 of packets, and each simulation run lasts for 10^6 time slots, a half of which is the warmup period in order to obtain stable statistics.

In the following, we will present the simulation results on different properties of the algorithms. In the legend of the figures, we append "-P" to the name of the scheduling algorithm if it is pipelined, and append "-PH" if HOL rescheduling is considered.

4.1 Analytical Convergence Result

The convergence property of PIM was analyzed in [3], and an average number of convergence iterations $\log_2 N + 4/3$ was obtained. Since then, $\log_2 N + 4/3$ has been commonly viewed as an estimation of the convergence iterations of iterative matching algorithms [1]. In Section 2, we also proved that the average number of convergence iterations of PIM2 is less than $\ln N + e/(e-1)$, which is a smaller number for any $N > 1$. We show in the following by simulation that, firstly, PIM and PIM2 have very similar convergence properties, and secondly, our analyzed result of the average convergence iterations is more accurate, in the sense that it is closer to the simulated data. As a result, if an iterative matching algorithm is designed to run with a fixed number of iterations, which is the case for most practical scheduling algorithms, $\ln N + e/(e-1)$ iterations are sufficient for the algorithm to converge in most cases.

In the simulations, we consider switch sizes of 16×16, 32×32, and 64×64, all of which have 100% Bernoulli or

burst uniform traffic. We look at the average convergence iterations of both PIM and PIM2, and compare them with the analysis results in this paper and in [3].

Figure 9(a) shows the simulations under Bernoulli uniform traffic. As can be seen, PIM and PIM2 have almost the same average convergence iterations for all the switch sizes. On the other hand, our analysis result $\ln N + e/(e-1)$ is closer to the simulation result than the classical one $\log_2 N + 4/3$. Figure 9(b) shows the simulation results under burst uniform traffic, and similar conclusions can be drawn that PIM and PIM2 have almost identical convergence property, and that $\ln N + e/(e-1)$ is a more accurate estimation. It should be noted that because of the burst nature, the convergence iterations of both algorithms are smaller than those under Bernoulli arrival. This can be explained by the fact that under burst arrival, within a small time interval, the incoming packets of an input port are not uniformly distributed among all the virtual queues. Thus, the convergence occurs earlier.

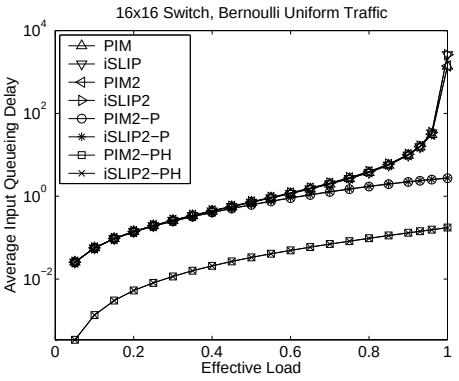
4.2 Input Queueing Delay

Input queueing delay is the interval from the time that a packet arrives at its input port to the time it is removed from the HOL of its virtual queue by the crossbar, i.e., the delay that a packet experiences in the input side of the switch.

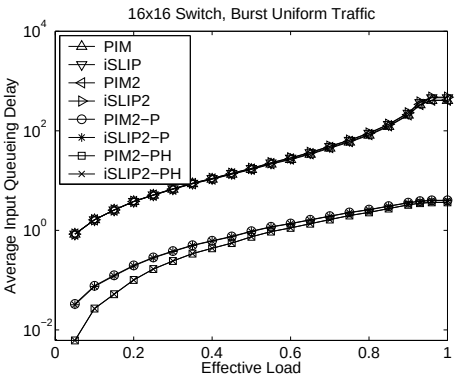
For the rest of the simulations, a 16×16 switch is considered. Figure 10(a) shows the input queueing delay of different algorithms under the Bernoulli uniform traffic. We notice that the four non-pipelined algorithms, i.e., PIM, iSLIP, PIM2, and iSLIP2, have almost the same input queueing delay. This is consistent with our analysis that two step algorithms and three step algorithms have similar performance. As can be easily seen, the pipelined two step iterative matching algorithms with speedup greatly shorten the average input queueing delay compared with the non-pipelined algorithms. In addition, with the HOL rescheduling mechanism, the delay is further reduced. Even under 100% load, the average input queueing delay for the pipelined PIM2 and iSLIP2 algorithms with HOL rescheduling is less than one time slot, which means that most packets are immediately transmitted to their output ports at the same time slot that they arrive. Thus, we can expect the two algorithms to exhibit similar performance to an OQ based scheduling algorithm. It is also interesting to note that, although PIM and iSLIP use different arbitration logic, they and all of their variants have very similar performance.

The simulation results under the burst uniform traffic are given in Figure 10(b). It is clear that the pipelined algorithms still achieve shorter input queueing delay than the non-pipelined algorithms. Although the delay of PIM2-PH and iSLIP2-PH are still shorter than that of PIM2-P and iSLIP2-P, the HOL rescheduling mechanism is less effective in reducing the input queueing delay than that under the Bernoulli uniform traffic. This can be explained by the burst nature of the traffic. Under burst arrival, the incoming packets of adjacent time slots have bigger chances to destine to the same output port. Therefore, if the SOL of a virtual queue is empty, it is very likely that its HOL is empty as well. Thus the HOL rescheduling mechanism cannot contribute much in this situation.

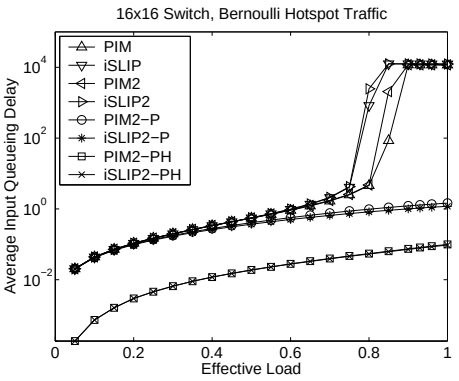
Figure 10(c) and (d) plot the results under the Bernoulli hotspot traffic and the burst hotspot traffic, respectively. Similar conclusions can be drawn that the pipelined algorithms greatly reduce the input queueing delay and the HOL rescheduling mechanism can further lower it. However, since



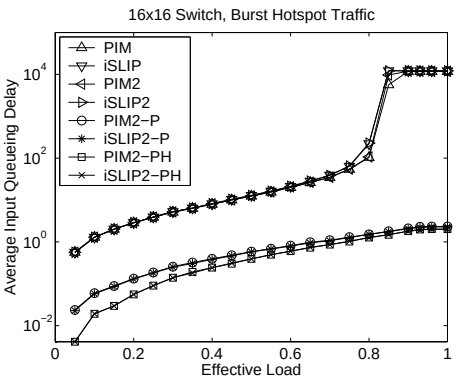
(a) Bernoulli uniform traffic



(b) Burst uniform traffic



(c) Bernoulli hotspot traffic



(d) Burst hotspot traffic

Figure 10: Comparison of input queueing delay of different scheduling algorithms.

the packets are no longer uniformly distributed among all the output ports, the input queueing delay of the non-pipelined algorithms increases dramatically when the effective load is approaching 80%. Especially, under the Bernoulli hotspot traffic, iSLIP and iSLIP2 saturate earlier than PIM2, and PIM2 saturates earlier than PIM.

4.3 Transmission Delay

In a similar way, we can define output queueing delay as the interval from the time that a packet is transmitted by the crossbar to the output port to the time that it is delivered to the outline. Then transmission delay is the sum of the input queueing delay and output queueing delay, or the total time that a packet stays in the switch. For a VOQ switch, its packet transmission delay is equal to its input queueing delay, since packets transmitted to the output ports are immediately delivered to the outline. For an OQ switch, its input queueing delay is zero, and its packet transmission delay is equal to the output queueing delay. It is known that OQ switches achieves the shortest average packet transmission delay. As an ultimate performance benchmark criteria, a simple FIFO scheduling algorithm (OQFIFO) on OQ switches is also included in the following comparison.

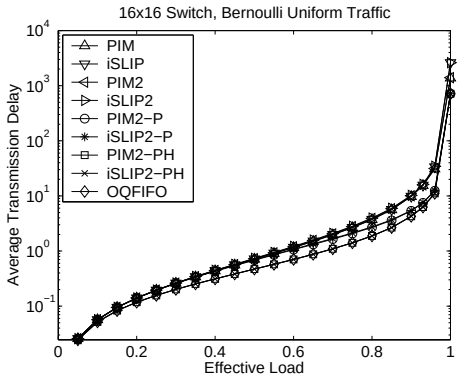
The average transmission delay of the various algorithms under the Bernoulli uniform traffic is given in Figure 11(a). We see that the pipelined algorithms achieves shorter delay than the non-pipelined ones. Furthermore, the transmission delay of the pipelined algorithms with HOL rescheduling is almost identical to that of OQFIFO, which has the shortest transmission delay. Figure 11(b) shows the situation under the burst uniform traffic. Due to the burst nature, the delay of all the algorithms is larger than that under Bernoulli arrival. Figure 11(c) and (d) show the results under the Bernoulli and burst hotspot traffic, respectively. Because the hotspot traffic is not uniformly distributed, the non-pipelined algorithms become unstable as the load approaches 80%. On the other hand, the pipelined algorithms are guaranteed to achieve 100% throughput.

4.4 Convergence Property

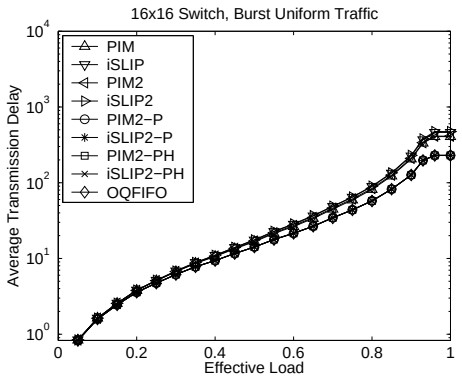
For an iterative matching algorithm, the average number of convergence iterations is a very important property, since an algorithm with smaller convergence iterations needs shorter scheduling time, and can achieve higher speed switching. As can be seen from Figure 12, all the algorithms use a similar number of iterations to converge. Generally, pipelined algorithms have smaller average convergence iterations than non-pipelined algorithms, because they find two matchings in each time slot. The HOL rescheduling mechanism incurs a slight increase in average convergence iterations. For iSLIP and iSLIP2, when the effective load of the Bernoulli uniform traffic approaches 100%, their convergence iterations decrease due to the round robin pointer desynchronizing mechanism.

5. CONCLUSIONS

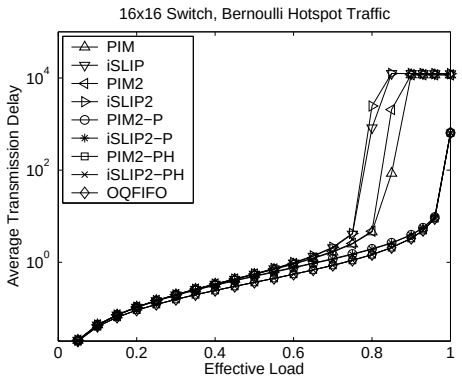
In this paper, we have studied two step iterative matching algorithms for VOQ switches and CIOQ switches. First, we analyzed the advantages of the two step iterative matching algorithms and presented Two Step Parallel Iterative Matching (PIM2) as an example algorithm for VOQ switches. We theoretically proved that the average number of convergence iterations of PIM2 is less than $\ln N + e/(e-1)$, and showed



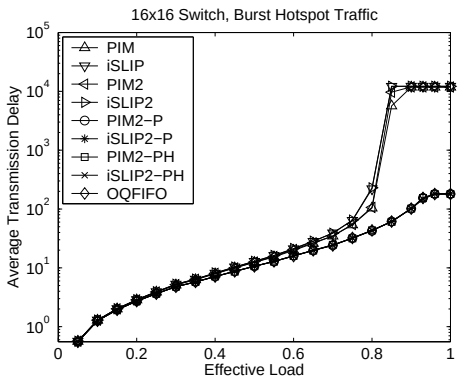
(a) Bernoulli uniform traffic



(b) Burst uniform traffic

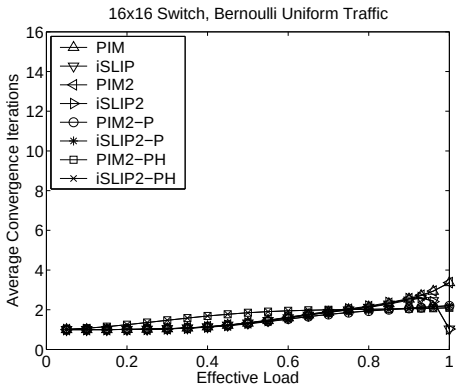


(c) Bernoulli hotspot traffic

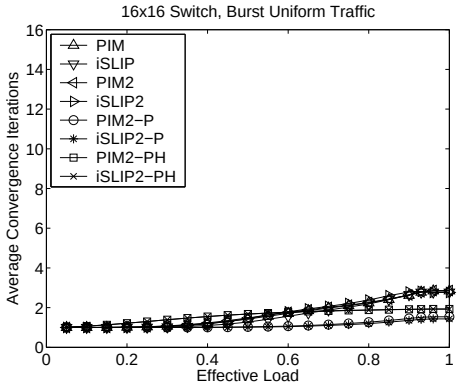


(d) Burst hotspot traffic

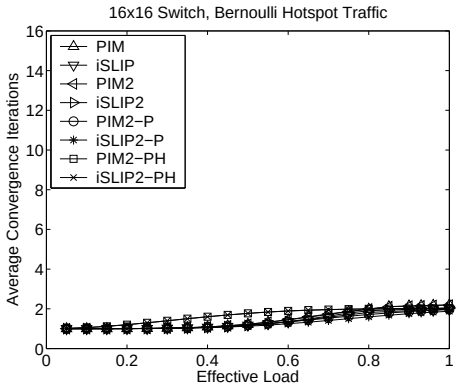
Figure 11: Comparison of transmission delay of different scheduling algorithms.



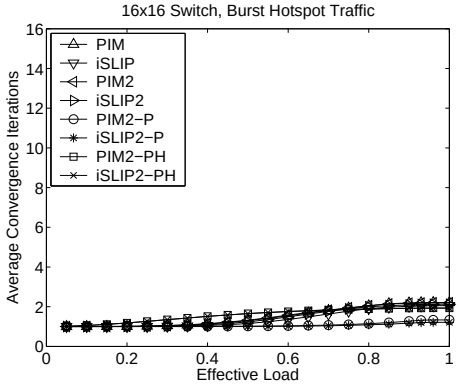
(a) Bernoulli uniform traffic



(b) Burst uniform traffic



(c) Bernoulli hotspot traffic



(d) Burst hotspot traffic

Figure 12: Comparison of convergence iterations of different scheduling algorithms.

through simulations that it is a more accurate estimation than the classical result $\log_2 N + 4/3$ in [3]. We also proposed a scheme called Second of Line (SOL) matching to efficiently pipeline two step iterative matching algorithms for CIOQ switches. The scheme does not require extra arbitration hardware and can make two scheduling decisions in each time slot. More importantly, any two step iterative algorithm pipelined with the proposed scheme is guaranteed to achieve 100% throughput for any admissible traffic. In order to further reduce the packet delay, the HOL rescheduling mechanism was proposed to improve the matching chances of the SOL matching. Extensive simulations were also conducted to test the performance of different scheduling algorithms. The simulation results show that the pipelined two step iterative matching algorithms are stable under both uniform and non-uniform traffic, and the HOL rescheduling mechanism enables the algorithms to achieve the same packet transmission delay as an OQ based scheduling algorithm.

6. REFERENCES

[1] N. McKeown, "The iSLIP scheduling algorithm for input-queued switches," *IEEE/ACM Trans. Networking*, vol. 7, no. 2, pp. 188-201, 1999.

[2] N. McKeown, A. Mekkittikul, V. Anantharam and J. Walrand, "Achieving 100% throughput in an input queued switch," *IEEE Trans. Commun.*, vol. 47, no. 8, pp. 1260-1267, 1999.

[3] T. Anderson, S. Owicki. J. Saxe and C. Thacker, "High-speed switch scheduling for local-area networks," *ACM Trans. Comput. Syst.*, vol. 11, no. 4, pp. 319-352, Nov. 1993.

[4] M. J. Karol, M. J. Hluchyj and S. P. Morgan, "Input versus output queueing on a space-division packet switch," *IEEE Trans. Commun.*, vol. 35, pp. 1347-1356, 1987.

[5] Nick McKeown, "A fast switched backplane for a gigabit switched router," *Business Communications Review*, vol. 27, no. 12, 1997.

[6] J. G. Dai and B. Prabhakar, "The throughput of data switches with and without speedup," *IEEE INFOCOM '00*, vol. 2, pp. 556-564, Tel Aviv, Israel, Mar. 2000.

[7] N. McKeown, M. Izzard, A. Mekkittikul, B. Ellesick and M. Horowitz, "The tiny tera: a packet switch core," *IEEE Micro,* vol. 17, no. 1, pp. 26-33, Feb. 1997.

[8] C. Partridge et al., "A 50-Gb/s IP router," *IEEE/ACM Trans. Networking,* vol. 6, no. 3, pp. 237-248, 1998.

[9] N. McKeown, "Scheduling algorithms for input queued cell switches," PhD Thesis, University of California at Berkeley, May 1995.

[10] J. Hopcroft, R. Karp, "An $N^{5/2}$ algorithm for maximum matching in bipartite graphs," *SIAM Journal of Computing*, vol. 2, no. 4, pp. 225-231, Dec. 1973.

[11] R. Tarjan, "Data structures and network algorithms," *CBMS-NSF Regional Conference Series in Applied Mathematics*, Dec. 1983.

[12] R. Panigrahy, A. Prakash, A. Nemat, and A. Aziz, "Weighted Random Matching: a simple scheduling algorithm for achieving 100% throughput," *Proc. of the IEEE Workshop on High Performance Switching and Routing*, pp. 111-115, Phoenix, AZ, 2004.

[13] D. Pan and Y. Yang, "FIFO based multicast scheduling algorithm for VOQ packet switches," to appear in *IEEE Trans. Computers*, vol. 54, no. 10, October 2005.

[14] Y. Li, S. S. Panwar and H.J. Chao, "Exhaustive service matching algorithms for input queued switches," *Proc. of IEEE Workshop on High Performance Switching and Routing*, pp. 253-258, 2004.