Hardware Efficient Two Step Iterative Matching Algorithms for VOQ Switches

Deng Pan
Dept. of Computer Science
State University of New York
Stony Brook, NY 11794, USA

Yuanyuan Yang
Dept. of Electrical and Computer Engineering
State University of New York
Stony Brook, NY 11794, USA

Abstract

Virtual output queued (VOQ) crossbar switches have been demonstrating advantages as high speed interconnects. They eliminate the Head of Line (HOL) blocking, which limits the maximum throughput of single input queued switches, and do not require switching fabrics with speedup capability, which prevents output queued switches from being cheaply implementable. Existing practical VOQ scheduling algorithms work in an iterative manner, and each iteration usually includes three steps: request, grant and accept. By incorporating arbitration into the request step, the accept step can be eliminated, and two step iterative matching can be achieved. While two step algorithms achieve almost identical performance as three step algorithms, they have extra advantages, such as simpler hardware implementation, shorter scheduling time, and less data exchange. As examples of two step iterative matching algorithms, we present Two Step Parallel Iterative Matching (PIM2) and Two Step iSLIP (iSLIP2), and theoretically analyze the convergence property of PIM2. Furthermore, because the request step and grant step perform similar operations, and the two steps always progress in a sequential manner, we propose a hardware efficient implementation for two step iterative matching algorithms which requires only one set of arbitration logic. We conduct extensive simulations, and the results demonstrate 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 that two step algorithms and three step algorithms have almost identical performance.

Keywords: Scheduling, virtual output queued switch, iterative algorithms, convergence, crossbars.

I. Introduction

Crossbar switches are widely used as high speed interconnects in different computing environments, such as PC clusters, Internet routers, and system-on-chip networks. With the requirement of high throughput and cheap implementation, the virtual output queued (VOQ) switch with a crossbar switching fabric has become the preferred structure for high speed switching [1] - [5], which operates with fixed length packets in a synchronous time slot mode. The structure of an $N \times N$ VOQ crossbar switch is illustrated in Figure 1. Input ports and output ports are connected by a crossbar switching fabric. The crossbar has non-blocking switching capability and can remove one packet from each input port and deliver one packet to each output port in every time slot. Temporarily blocked packets are buffered at the input side using the VOQ buffering strategy, i.e., each input port has N (logically) separate queues to buffer the packets to the N different output ports. Thus, a packet will not be held up by another packet ahead of it that goes to a different output port.

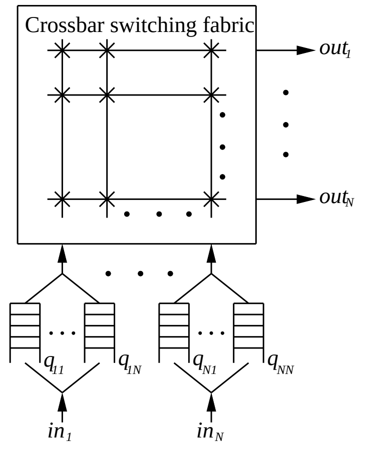


Fig. 1. For an $N \times N$ VOQ crossbar switch, blocked packets are buffer at the input side, and each input port has N queues to buffer the packets to the N different output ports.

Several advantages of the VOQ crossbar switch make it more attractive than other structures. First of all, the crossbar provides non-blocking capability, which is necessary for achieving high speed switching. Also, the time slot work mode significantly simplifies the design of the switch, and accelerates the scheduling and switching processes. Comparing with the output queued (OQ) switch, the VOQ switch does not require an N speedup crossbar and is cheap to implement. Since the OQ switch buffers packets only at the output side, if the packets arriving at different input ports are destined to the same output port, all the packets must be transmitted simultaneously. Therefore, the switching speed of the internal fabric must be N times faster than the sending speed of the input port. On the other hand, comparing with the single input queued (SIQ) switch, the VOQ switch removes the HoL blocking using the VOQ buffering strategy. For the SIQ switch, each input port has a single queue to buffer all the incoming packets. If the head of line (HoL) packet is blocked, all the packets behind it cannot be scheduled to transmit even thought their destination output ports may be free. This is called the HoL blocking, which limits the maximum throughput of the SIQ switch to only about 58.6% [6].

Scheduling packets to be transferred from input ports to output ports to ensure low latency and high throughput is a challenging task. The scheduling problem on VOQ switches can be viewed as a special case of the bipartite graph matching problem, where input ports and output ports are the two disjoint sets of vertices and the edges between input ports and output ports are the scheduling decisions. Traditional scheduling algorithms, such as maximum size matching (MSM) [7] and maximum weight matching (MWM) [7], were designed to maximize the throughput of the switch. However, both MSM and MWM have high time complexity, which is $O(N^{2.5})$ [8] and $O(N^3 \log N)$ [9], respectively, and therefore are impractical for high speed implementation. Besides, 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 [10].

High speed switching imposes a requirement for high speed scheduling as well. As a result, iterative matching algorithms, such as PIM [2] 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 request to every output port for which it has a buffered packet.

Grant step. An output port grants one request among all the requests that it receives.

Accept step. An input port accepts one grant among all the grants that it receives. 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 in each time slot by incrementally adding input-output pairs, without removing the ones made earlier. In general, a maximal matching is easier to obtain but may be smaller than a maximum matching, which has the globally largest size or weight.

In this paper, we discuss two step iterative matching algorithms for VOQ switches, and propose a hardware efficient implementation of the algorithms. First, we show that by incorporating arbitration into the request step, the accept step can be eliminated, and thus two step iterative matching can be achieved, as shown in Figure 3. The advantages include simpler implementation, shorter scheduling time, and less data exchange. As examples, we present Two Step Parallel Iterative Matching (PIM2) and Two Step iSLIP (iSLIP2), and theoretically analyze the convergence property of PIM2. Furthermore, because the request step and grant step of a two step iterative matching algorithm have similar functionality, and the two steps never work at the same time, the algorithm can be implemented in a hardware efficient manner with only one set of arbitration logic. Finally, extensive simulations are conducted, and the results demonstrate that our analytical result, $\ln N + e/(e-1)$, is a more accurate estimation of the average convergence iterations for iterative matching algorithms than the classical result, $\log_2 N + 4/3$, and the results also demonstrate that two step and three step iterative matching algorithms have very similar performance.

The rest of this paper is organized as follows. Section II presents the two step iterative matching algorithms for VOQ switches. Section III proposes a hardware efficient implementation of the algorithms. Section IV gives the simulation results. Finally, Section V concludes the paper.

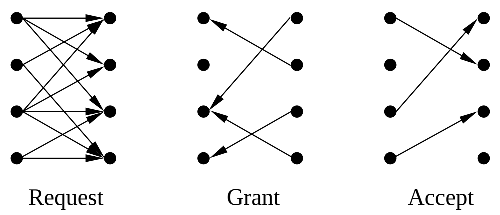


Fig. 2. Three step iterative matching.

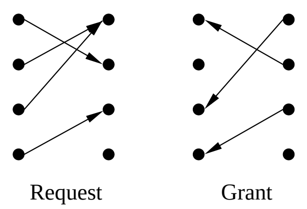


Fig. 3. Two step iterative matching.

II. Two Step Iterative Matching Algorithms

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 $\ln N + e/(e-1)$. We also discuss other generalizations of two step iterative matching algorithms in this section, including Two Step iSLIP (iSLIP2) and FIFOMS.

A. *Advantages of Two Step Iterative Matching*

In one iteration of a three step iterative matching algorithm, 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 and two step iterative matching is achieved, as illustrated in Figure 3.

Comparing with existing three step algorithms, two step iterative matching has the following advantages. Firstly, by eliminating the accept step, the time for one iteration of the algorithm is reduced, and thus shorter total scheduling time is needed. Secondly, for a two step iterative matching algorithm, the request step and the accept step are carried out in the same way, i.e., to arbitrate among N candidates 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 during the scheduling process are greatly reduced. Especially, since each input port can receive at most one grant, the request step of the next iteration can start immediately after the only grant is received. While in 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.

On the other hand, it can be expected that two step algorithms and three step algorithms have similar performance. 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, which is easy to see when comparing Figure 2 and Figure 3.

B. 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 [2]. 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. Then 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 to any output port, and fairness is achieved.

C. Convergence Property of PIM2

In this subsection, we discuss the convergence property of PIM2. 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 need more iterations or longer time to converge, which is of course unfavorable for a practical scheduling algorithm. However, our following theoretical analysis and the simulation results in Section IV both show that three step algorithms and two step algorithms have almost identical convergence properties.

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 port has packets to it. We define the following terms.

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;

$fanout(in_i) = \{out_j | out_j \text{ is free, and } q_{ij} \text{ is not empty}\}$;

$FreeIn(k) = \{in_i | in_i \text{ is free after } k \text{ iterations}\}$;

$O(FreeIn(k)) = \bigcup_{in_i \in FreeIn(k)} fanout(in_i)$;

$p(k) = \min\{|FreeIn(k)|, |O(FreeIn(k))|\}$.

$p(k)$ 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.,

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

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 other 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 discuss two possible cases.

Case 1: $m \geq n$ 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 value of n, say, $n \leq 10^6$, $f(n) \leq \frac{1}{e}$. Thus, we can obtain

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

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 $m \geq 2$.

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 $x \neq 0$; 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

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

Thus, in both cases, we have $E(p(k+1)) \le p(k)/e$. ■

Lemma 2: For an $N \times N$ switch, the expected value of the potential after k iterations is less than or equal to N/e^k, i.e.,

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

Proof. We prove it by induction.

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

Inductive step: Suppose $E(p(k)) \le N/e^k$ holds. From Lemma 1, we know that $E(p(k+1)|p(k)) \le p(k)/e$, or $\sum_{i=0}^{N} i \Pr\{p(k+1) = i | p(k) = j\} \le j/e$. Then,

$$\begin{aligned}
&E(p(k+1))\\
&= \sum_{i=0}^{N} i \Pr\{p(k+1) = i\}\\
&= \sum_{j=0}^{N}\sum_{i=0}^{N} i \Pr\{p(k+1) = i | p(k) = j\} \Pr\{p(k) = j\}\\
&= \sum_{j=0}^{N} \Pr\{p(k) = j\} \left(\sum_{i=0}^{N} i \Pr\{p(k+1) = i | p(k) = j\}\right)\\
&\le \sum_{j=0}^{N} \Pr\{p(k) = j\} \frac{j}{e} = \frac{1}{e}\sum_{j=1}^{N} j \Pr\{p(k) = j\}\\
&= \frac{E(p(k))}{e}
\end{aligned}$$

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

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

Theorem 1: For an $N \times N$ switch, the average number of convergence iterations of PIM2 is less than or equal to $\ln N + e/(e-1)$, i.e.,

$$E(C) \le \ln N + \frac{e}{e-1}$$

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

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

On the other hand,

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

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

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

■

The convergence property of PIM was also analyzed in [2], and an average number of convergence iterations, $\log_2 N + 4/3$, was obtained. We will show by simulation in Section IV that our result, $\ln N + e/(e-1)$, is a more accurate estimation for the convergence iterations of iterative matching algorithms.

D. 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 [2] 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. The two step version of iSLIP, which we call iSLIP2, can be described 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, and it is possible for the algorithm to converge with one iteration. 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. A similar algorithm called Dual Round Robin Matching (DRRM) was proposed in [11] with a different method to update the round robin pointers.

Two step iterative matching can also be used to schedule multicast traffic, such as FIFOMS in [12]. 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 [12], the two step multicast iterative matching algorithm has small average convergence iterations and achieves short multicast latency as well.

III. Hardware Implementation

Practical scheduling algorithms are expected to be able to efficiently implement in hardware to make fast decisions for high speed switching. In this section, we propose a hardware efficient

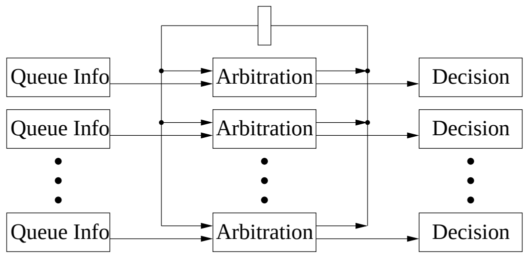


Fig. 4. High level implementation diagram of two step iterative matching algorithms.

implementation for two step iterative matching algorithms. The implementation requires only one set of arbitration logic, which is alternatively used for request arbitration and grant arbitration.

For a two step iterative matching algorithm, its request step and grant step perform similar operations. In other words, in each step, there are up to N candidates (N output ports as request candidates in the request step and N requests as grant candidates in the grant step), and arbitration is made to select one from the N candidates. On the other hand, the two steps never work at the same time, but instead progress in a sequential manner. The reason is that only free input ports can send requests, but whether an input port is free or not is not known until the grant step of the last iteration finishes. As can be seen, the request arbitration logic and grant arbitration logic are only busy for a half of the total time and are not fully utilized.

Based on the above observations, we propose a hardware efficient implementation for two step iterative matching algorithms, which needs only one set of arbitration logic. Figure 4 shows the high level diagram of the implementation. At the beginning of each time slot, the input of the scheduling algorithm is initialized with the virtual queue occupancy of each input port. Then, arbitration is made based on the queue information to send a request on behalf of each input port, which is fed back to the arbitration logic. A second round of arbitration is made with the request information to generate grant for each output. At the end of one iteration, matching results are saved in the decision registers, and will be finally forwarded to the crossbar as control signals to transmit the scheduled packets.

Figure 5 shows the details inside the arbitration logic. Since the hardware is alternatively used in the request step and grant step, there is a signal R/G to indicate the current working state. When R/G is 0, the algorithm is in the request step. If the virtual queue of an input port has buffered packets and its corresponding output port has not been matched, it sends a signal to the arbiter as a candidate for the request arbitration. According to a specific arbitration rule, the arbiter chooses one from the up to N candidates and the result is fed back as the input of the arbiters. Next, the algorithm enters the grant step and the R/G signal becomes 1. The requests received in the previous step are sent to corresponding arbiters, and each arbiter chooses one to grant. Thus, one iteration of the algorithm has completed, and the grant results are sent to the matching decision registers.

Although the actual arbitration rules used in the request step and grant step may vary among different scheduling algorithms, they are usually very similar in the two steps of the same algorithm. For example, in both steps of PIM2, the arbiter acts as a random selector, and randomly chooses one candidate. In the

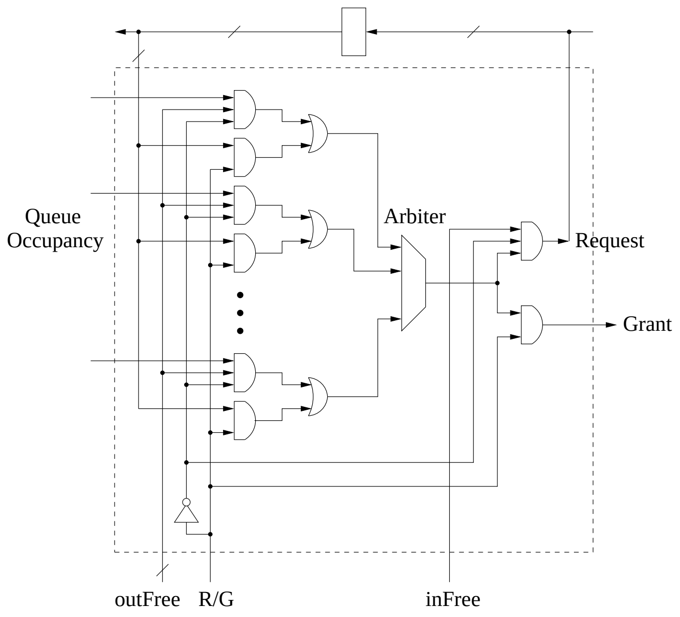


Fig. 5. Details of the arbitration logic.

two steps of FIFOMS, the arbiter is actually a comparator, and it compares the time stamps of the candidates and chooses the smallest one. For iSLIP2, the arbiter is a priority encoder, and picks the candidate appearing next to the corresponding round robin pointers. Because iSLIP2 uses different round robin pointers in different steps, each arbiter needs two registers to save the current positions of the two different round robin pointers.

IV. Simulation Results

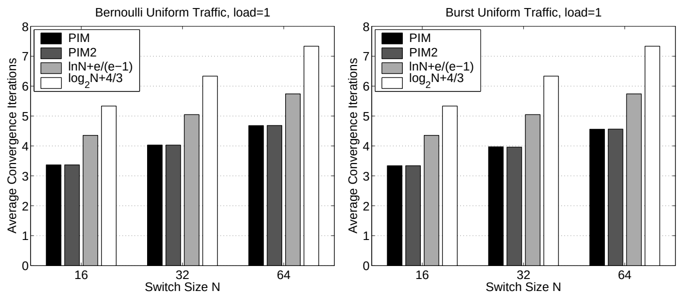
In this section, we conduct simulations to verify the accuracy of the convergence iteration analysis in Section II, and also to compare the performance of two step algorithms and three step algorithms.

Both Bernoulli arrival and burst arrival are considered in the simulation. Bernoulli arrival can be described by its average arrival rate p. In other words, each input port has the probability of p to have a new packet to arrive at the beginning of a time slot. 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 process can independently switch between off and on states. Burst arrival can be described using two parameters α and β. α is the probability to switch from the off state to the on state, or alternatively the average length of the off state is $1/\alpha$. β is the probability to switch from the on state to the off state, or the average length of the on state is $1/\beta$. Therefore, the average arrival rate is $\beta/(\alpha+\beta)$.

In the following, we will present the simulation results on different properties of the algorithms. Each simulation run lasts for 10^6 time slots, a half of which is the warmup period in order to obtain stable statistics.

A. Analytical Convergence Result

The convergence property of PIM was analyzed in [2], 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



(a) Bernoulli uniform traffic (b) Burst uniform traffic

Fig. 6. Comparison of average convergence iterations with different switch sizes.

an estimation of the convergence iterations of iterative matching algorithms, such as iSLIP [1], WPIM [13] and FEM [14]. In Section II, 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 almost identical convergence properties, and secondly, our analytical result on the average convergence iterations is more accurate, in the sense that it is closer to the simulation results. 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.

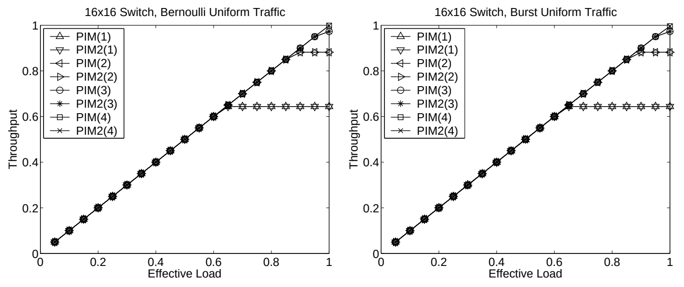
In the simulations, we consider switch sizes of 16×16, 32×32 and 64×64, all of which have 100% Bernoulli or burst uniform traffic. For uniform traffic, the destination of a new incoming packet is uniformly distributed among all the output ports. Denoting the arrival rate of q_{ij} by λ_{ij}, then $\lambda_{ij} = p/N$, where p is the load of the switch.

We look at the average convergence iterations of both PIM and PIM2, and compare them with the analytical results in this paper and in [2].

Figure 6(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 analytical result, $\ln N + e/(e-1)$, is closer to the simulation result than the classical result, $\log_2 N + 4/3$. Figure 6(b) shows the simulation results under burst uniform traffic, and similar conclusions can be drawn that PIM and PIM2 have almost identical convergence properties, 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 slightly 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, each input has fewer matching candidates, and the convergence occurs earlier.

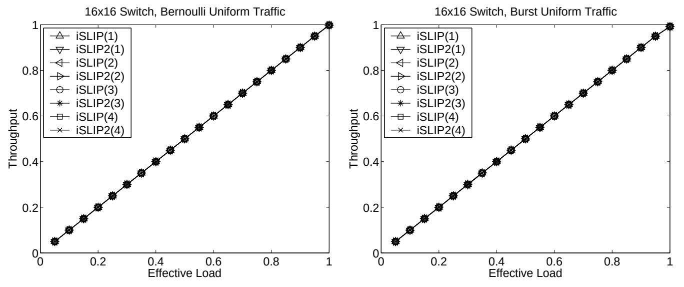
B. Throughput with Fixed Iterations

As mentioned above, practical scheduling algorithms usually run with a fixed number of iterations rather than run until converge, so that the scheduling time needed in each time slot is a constant. In the following, we examine the throughput of the scheduling algorithms with fixed iterations. For the rest of the simulations, a 16×16 switch is considered.



(a) Bernoulli uniform traffic (b) Burst uniform traffic

Fig. 7. Throughput of PIM and PIM2 with fixed iterations.

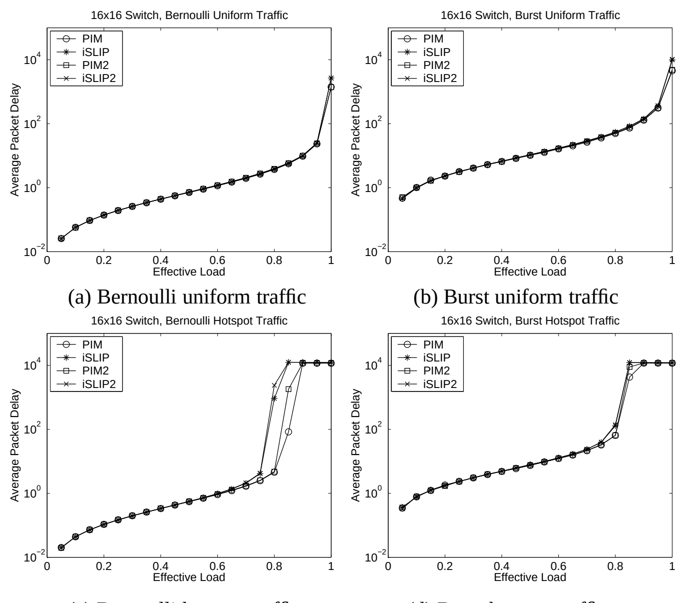


(a) Bernoulli uniform traffic (b) Burst uniform traffic

Fig. 8. Throughput of iSLIP and iSLIP2 with fixed iterations.

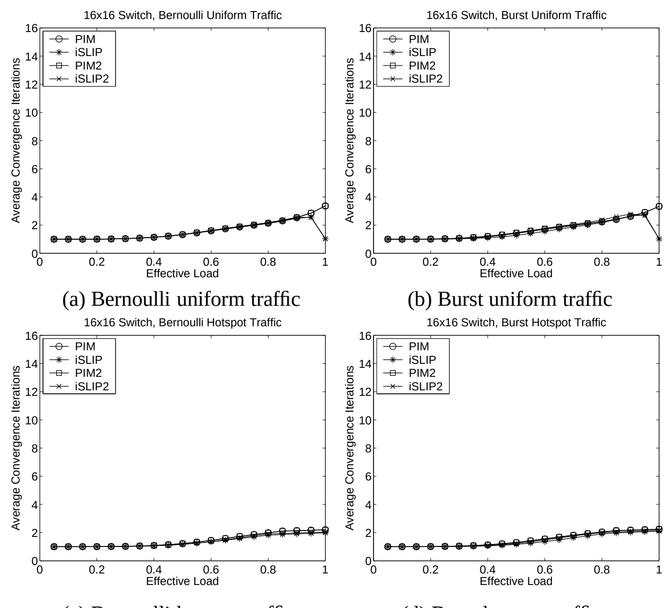
Figure 7(a) shows the throughput of PIM and PIM2 with 1, 2, 3 and 4 iterations under the Bernoulli uniform traffic. In the legend, the number in the brackets is the number of the fixed iterations. It can be noticed that, with the same fixed number of iterations, PIM and PIM2 have very similar throughput performance, and their maximum throughput increases as the number of iterations increases. To be more specific, with one iteration, PIM(1) and PIM2(1) can deliver all the incoming packets when the load is small, but saturate at about 64% throughput. With two iterations, PIM(2) and PIM2(2) significantly increase their maximum throughput to about 88%. When running with three iterations, the throughput of PIM(3) and PIM2(3) can be as high as 97%. Given one more iteration, the maximum throughput of PIM(4) and PIM2(4) increases to 99.9%, which means that they practically achieve 100% throughput. The results under burst uniform traffic are similar to those under Bernoulli uniform traffic, which are given in Figure 7(b). On the other hand, based on the analysis in Section II, we know that the throughput of PIM2 with i fixed iterations is approximately $1 - 1/e^i$, which is 63%, 86%, 95% and 98% when i is equal to 1, 2, 3 and 4, respectively. It can be seen that the simulation results are consistent with the theoretical analysis.

Figure 8(a) and (b) plot the throughput of iSLIP and iSLIP2 with fixed iterations under the Bernoulli uniform traffic and burst uniform traffic, respectively. Similarly, iSLIP and iSLIP2 do not have noticeable difference in throughput. On the other hand, because of the round robin desynchronizing mechanism and the uniformly distributed traffic, iSLIP and iSLIP2 achieve 100% throughput even with only one iteration. However, as indicated in Figure 10 of this section, the convergence iterations of iSLIP and iSLIP2 are usually larger than one unless the load is small or close to one. This means that, with one fixed iteration,



(a) Bernoulli uniform traffic (b) Burst uniform traffic (c) Bernoulli hotspot traffic (d) Burst hotspot traffic

Fig. 9. Comparison of average packet delay of different scheduling algorithms.



(a) Bernoulli uniform traffic (b) Burst uniform traffic (c) Bernoulli hotspot traffic (d) Burst hotspot traffic

Fig. 10. Comparison of average convergence iterations of different scheduling algorithms.

the algorithms may not converge under some circumstances, or there are still packets that can be scheduled to transmit but are postponed to the next time slot. Thus, although iSLIP and iSLIP2 can achieve 100% throughput with one iteration under uniform traffic, the packet delay may be significantly increased in this case.

C. Packet Delay

The transmission delay of a packet is the interval from the time that the packet arrives at its input port to the time it is removed from the head of its virtual queue by the crossbar. Since each output port receives at most one packet per time slot, the received packet can be immediately sent to the outline, and thus the transmission delay is the total time that a packet stays in the switch.

Figure 9(a) shows the average packet delay of different algorithms under the Bernoulli uniform traffic. As can be seen, the two step algorithms (PIM2 and iSLIP2) have almost identical performance as their corresponding three step algorithms (PIM and iSLIP), respectively. It also can be noticed that, although PIM and iSLIP use different arbitration rules, they and their two step counterparts have similar average packet delay. Figure 9(b) gives the simulation results under the burst uniform traffic, in which the four algorithms also exhibit similar performance. However, due to the burst nature, the delay of all the algorithms is longer than that under Bernoulli arrival.

The simulation is also conducted under non-uniform traffic, or hotspot traffic [15] in our case. 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 an equal amount of packets. In our simulations, we set $\lambda_{ii} = p/2$ and $\lambda_{ij} = p/2(N-1)$ for $i \neq j$. Figure 9(c) plots the simulation results under the Bernoulli hotspot traffic. While the arriving traffic is not uniformly distributed, all the four algorithms still give each virtual queue of an input port equal matching chance. As a result, when the effective load becomes large, the delay increases dramatically and finally reaches the limit allowed by the maximum queue length. Figure 9(d) gives the situation with the burst hotspot traffic. Again, the curves exhibit a similar trend as that in Figure 9(c), but the delay is longer than that under the Bernoulli hotspot traffic because of the burst nature.

D. 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 10, all the four algorithms need a similar number of iterations to converge in most cases. In Figure 10(a) and (b), when the effective load of the Bernoulli uniform traffic or burst uniform traffic approaches one, the convergence iterations of iSLIP and iSLIP2 decrease to one due to the round robin pointer desynchronizing mechanism. However, when the traffic is not uniformly distributed, as in Figure 10(c) and (d), iSLIP and iSLIP2 do not show significant advantages over PIM and PIM2.

V. Conclusions

In this paper, we have studied two step iterative matching algorithms for VOQ crossbar switches. By incorporating arbitration into the request step, the accept step in traditional three step iterative matching algorithms can be eliminated. While the two step iterative matching algorithms maintain almost identical performance as three step algorithms, they introduce extra advantages, such as simpler hardware implementation, shorter scheduling time, and less data exchange. As examples, we presented Two Step Parallel Iterative Matching (PIM2) and Two Step iSLIP (iSLIP2). We theoretically proved that the average number of convergence iterations of PIM2 is less than $\ln N + e/(e-1)$, and showed by simulation that it is a more accurate estimation than the classical result $\log_2 N + 4/3$ in [2].

Based on the characteristic that the request step and grant step have similar functionality and do not work at the same time, we proposed a hardware efficient implementation for the two step iterative matching algorithms. The implementation requires only one set of arbitration logic, which can be alternatively used for request arbitration and grant arbitration. Extensive simulations were also conducted to test the performance the two step iterative matching algorithms. The simulation results demonstrated that two step algorithms and three step algorithms have very similar performance.

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] 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.

[3] 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.

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

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

[6] 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.

[7] 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.

[8] 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.

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

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

[11] Y. Li, S.S. Panwar and H.J. Chao, "On the performance of a dual round-robin switch," *IEEE INFOCOM 2001*, pp.1688-1697, Anchorage, AK, Apr. 2001.

[12] D. Pan and Y. Yang, "FIFO based multicast scheduling algorithm for VOQ packet switches," *IEEE Trans. Computers*, vol. 54, no. 10, pp. 1283-1297, Oct. 2005.

[13] D. Stiliadis and A. Varma, "Providing bandwidth guarantees in an input-buffered crossbar switch," *Proc. IEEE INFOCOM '95*, pp. 960-968, Apr. 1995.

[14] H.J. Chao, K.-L. Deng and Z. Jing, "Petabit photonic packet switch (P^3S)," *IEEE INFOCOM '03*, San Francisco, Apr. 2003.

[15] 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.