Simulation of Iterative Matching for Combined Input and Output Queueing

Srinivasan Pichai, Sriram Mudulodu

{vasan,msriram}@leland.stanford.edu

May 1998

1 Abstract

Since its introduction the *Stable Marriage* problem has been a subject of interest in mathematics and computer science. Recently this algorithm has found application in the area of switch scheduling algorithms for high performance switches. Inputs and Output ports of the switch compute their preference lists based on expected departure times for an ideal output queued switch. The stable matching as computed by the Galey-Shapley for this set of preferences determines the configuration of the interconnection fabric. The nature of the stable matching enables the emulation of an output-queued switch with combined input and output queueing using a speedup factor of 2.

However it is important to compute the stable match efficiently for high performance. Hence parallel iterative versions of the algorithm have been proposed. In this report we investigate the convergence time of the parallel stable matching algorithm. The definition of the preference lists imposes special constraints on the problem and this reduces the worst case complexity of the algorithm. Simulations have shown that convergence time for the average case is also considerably lower than the general version of the algorithm. We also discuss the various implementation details of the algorithm. Our work is based on the algorithm presented in [CGMP98]. One of the advantages of this emulation is that the cell latency can be accurately controlled as it would be in an output-queued switch.

2 The Gale-Shapley Algorithm

The stable marriage problem was first introduced by Gale and Shapley in [GS62]. In the basic problem, a set of women and men all rank the members of the opposite sex according to their individual preferences. A matching is *unstable* if there is a man and a woman who prefer each other to their current partners. Any matching which is not unstable is defined to be stable. It can be proven that any arbitrary preference matrix admits to a solution. The original algorithm proposed by Gale and Shapley comes in two flavours - the *male optimal* and the *female optimal*. In the male optimal version an unmatched man proposes to the woman next on his preference list. If that woman is unengaged she accepts his proposal. If the woman is already engaged and if she prefers him to her current partner then she breaks off her engagement and accepts his proposal. If the woman rejects his proposal the man drops her from his preference list and goes on to his next choice. The algorithm is guaranteed to terminate with every man and woman engaged.

Note that a new proposal might result in a cascade of changes. This algorithm is different from traditional graph matching algorithms in that edges that are formed may be dropped at

a subsequent stage. Although the original problem considered the case of complete preference lists, the same algorithm works for incomplete preference lists. The instances of the stable marriage problem encountered in switch scheduling algorithms have this property. In such a case the algorithm terminates with a set of men and women who cannot be matched and a set of stable pairings. It can be shown that in the worst case the algorithm can take up-to $O(N^2)$ steps. Another interesting aspect of the algorithm is that the resulting match is male optimal in the sense that each man receives the best partner that he could have in *any* stable matching.

2.1 The Parallel version

A few researchers have proposed parallel versions of the Gale-Shapley algorithm. But some of them are complex and cannot be implemented in a high speed switch in hardware. We look at a simple parallel iterative version which is similar to the PIM (Parallel Iterative Matching) algorithm presented in [AOST93]. This algorithm proceeds in a sequence of iterations. In the first iteration all the men propose to the woman of their first choice. Each woman selects the best candidate from amongst the proposals that she receives. The men who remain unengaged advance to the next woman on their preference list and participate in the next round. It can be shown that this algorithm could still take $O(N^2)$ time in the worst case. However in the instances encountered in switch emulation it can be shown that the worst case complexity is $O(N)$. Moreover some bounds on the average performance can be obtained by relating this procedure to the serial algorithm .

3 Average Performance

We give a brief summary of the analysis presented in [Knu76] Although the worst case complexity is $O(N^2)$ it has been shown in [Knu76] that the average case takes $O(N \log N)$ steps. We present a brief description of the proof presented in [Knu76]. The problem is reduced to the well known coupon collection problem. Consider for a moment that the women's preference matrix is fixed. A sequence consisting of the women's names can be mapped to particular instance of the men's matrix and the length of the sequence corresponds to the number of steps in the serial algorithm. The condition for this to occur is that the sequence should contain all the name of the women. This is the classic coupon collection - In a single trial , each type of coupon is equally likely to be drawn. It can be shown that the average number of independent trials to obtain all coupons is related to the harmonic sum H_n. Hence we expect in the average case the number of steps is $O(N \log N)$.

This analysis can be related to the parallel version in the following way. A consequence of the above analysis is that on the average each man makes about $\log N$ proposals. However the number of times a woman changes partners is lesser than this value. If a woman receives k proposals, we can make a rough approximation of the average number of times she changes partners by observing that she is likely to settle on the ith proposal with equal probability. The number of changes given k is roughly $O(\log k)$. Since k is $O(\log N)$ the total number of change of partners made by a woman is $O(\log \log N)$. In the parallel version if no woman changes her partner during an iteration, the current matching can be shown to be stable. Hence we can conjecture that the number of iterations is upper bounded by $O(N \log \log N)$ for the average case of the parallel algorithm. As we will see later the performance can be improved considerably

4 Stable Matching and Output Queue Emulation

In this section we discuss in detail the implementation of the output queue emulation presented in [CGMP98]. In this model the crossbar fabric and the output queues utilize a *speedup* factor to improve performance. Traditional output queued switches are able to transfer all the cells with the same destination within a time slot to the corresponding output. In contrast in an input queued switch only one cell can be delivered to a particular tagged output in a single time slot. This causes the phenomenon of HOL blocking - a cell at the head of the line which loses in the current slot prevents other cells behind it from getting access to the crossbar. Various proposals have been made to improve the performance of input queued switches.

Recently it has been shown that a moderate speedup of S for a combined input output queued switch results in a high overall throughput. A switch with a speedup of S can remove up-to S packets from each input port and transfer up-to S packets to a particular output port. We can consider the speedup of pure output queued switches to have a speedup of N. (where the switch is of size $N\ x\ N$). In contrast input-queued switches have a speedup of just 1. It has been proven in [CGMP98] that a speedup of $2-1/N$ is necessary and sufficient to emulate an output queued switch. This means that if the same input traffic is applied to the CIOQ switch and the shadow output queued switch the departure process for the cells should be indistinguishable. Note that no restrictions are placed on the arrival patterns of the input traffic. This is a general approach as compared to the approach in [CPK94] where it is shown that switch speedup can be used to increase the throughput for certain traffic patterns.

In our discussion we will assume that the speedup is exactly two. The input ports

maintain a preference list based on the cells waiting at that input. The output ports also maintain a preference list based on the cells which are destined to themselves. The proof of the algorithm relies on the definition of the *output cushion, time of departure, input thread and slackness* of an individual cell.

- *Time of Departure*(TOD) is the time that a cell c would have left the shadow output queued switch that we are emulating.
- *Output Cushion*(OC) of a cell c is the number of cells waiting at cell c's output port (in the CIOQ switch) which have a lower time of departure than cell c.
- The Input Thread (IT) of a cell is the number of cells ahead of it in its input priority list.

Intuitively, if the output cushion is small or zero the scheduling algorithm should try to deliver the cell urgently when its time of departure is reached. Alternately it is undesirable to have a large input thread which is approaching its time of departure. The output cushion may increase if more urgent cells destined for that output. The input thread decreases for a cell if a cell ahead of it departs during the current slot and it increases if a newly arrived cell is placed ahead of it in the input priority list. We should avoid a large input thread for a cell with a small output cushion. This motivates the definition of slackness.

- The slackness of a cell is the value of the input thread minus the value of the output cushion.

Slackness indicates the relative urgency of a cell after adjusting for its input thread and output cushion. The proof is based on the observation that the slackness is non-decreasing from time slot to time slot. Moreover if the initial slackness is non-negative the cell is guaranteed to reach its output before or at its time of departure.

The CIOQ switch functions in four phases. In the *arrival* phase the newly arrived cells are placed in an appropriate position in the input queues. In the first scheduling phase the cells with slackness zero participate and the stable match is used to establish the transfer matrix. In the departure phase one cell departs from every non-empty output queue. In the second scheduling phase again the cells of slackness participate in the stable matching and matched cells are transferred. The reason behind considering only cells of slackness zero is that this reduces the number of iterations for convergence of the parallel stable matching algorithm. It is shown in [CGMP98] using a dependency graph that the number of iterations cannot exceed N. The input preferences are based on the order in which cells occur in the input queue. If several cells destined to the same output want to participate it is enough to

consider the one that occurs first in the input priority queue. The output preferences are based on the time of departures. It is important to understand that if a cell participates in the stable matching its slackness is guaranteed to increase. In other words either a cell which is more urgently desired at its output or input is transferred.

5 Our implementation

We have simulated this algorithm using the SIM simulator package. The simulation proceeds in terms of slotted time. The input queues use VOQs to store cells destined for different outputs separately. At each input port we maintain a relative ordering of the different non-empty VOQs at that port. When a cell for a non-empty VOQ arrives the output port number has to be inserted in the $voqArray$ which maintains the relative ordering of the VOQs. One of the advantages of maintaining VOQs is that only the cells at the heads of the VOQs need to be considered. We also time-stamp a newly arrived cell with its projected time of departure in the shadow output queued switch.

During each scheduling phase we have to determine the output cushion of the first cell in each VOQ. This could be a potentially costly operation. To reduce the cost of this operation the output queue would have to be sorted. But this entails some complexity at the end of each transfer. We should also observe that this computation might have to be done at a particular output for several cells. Hence this could be a potential problem for a hardware implementation. Once we find the output cushion we can index into the VOQ ordering array for this input and compute its input thread. Thus the slackness can be determined using these two values. Another problem in a practical implementation would be the need to serialize the computation of the slackness for different VOQs at a particular input. This is so as we have to traverse the VOQs in the order dictated by the $voqArray$. To avoid this we would have to maintain the cumulative sum of the lengths of the VOQs but many of these would have to be updated on the arrival or transfer of a cell. Since we consider cells of slackness zero at each scheduling phase every cell is guaranteed to have a slackness of at least 1 before the next arrival phase.

Once we obtain the input and output preferences these are passed to the stable matching algorithm. In each a list of participants is obtained. Due to the incomplete nature of the preference lists some men and women may have small preference lists. An input can participate in the current round if its unmatched and if it has some preferences left in his list. The algorithm stops when the number of a participants in the round becomes zero. We recorded the number of iterations required for convergence during each scheduling phase.

6 Results and Observations

Somewhat surprisingly we found that the number of iterations is rather much lower than the worst case bound of $O(N)$. For instance even for a 64 x 64 switch we found that the number of iterations was lesser than four during a simulation period of 1,00,000 time slots. The iteration count and the frequency is presented below for different switch sizes. Interestingly the number of iterations was quite insensitive to the arrival patterns of the traffic. We set some of the inputs to produce bursty traffic, some produced Bernoulli traffic. The utilization values of the input ports also did not increase the number of iterations beyond four. The table below gives the distribution of the iterations and the corresponding frequency. For a 32 x 32 switch the majority of the scheduling phases require only one or two iterations. The length of the simulation also did not affect the results in any significant manner. Out

Iterations	Frequency
1	21191
2	27335
3	1455
4	19

Table 1: Iterations for 32 x 32 switch

of 50,000 schedulings only 1474 take more than 2 iterations which is less than 3% of the iterations. A similar experiment for a switch with 64 inputs and outputs was conducted. The results were still low. There was a slight increase in the number of schedulings of 4 iterations but large increase in the number of schedulings with iterations of value 2. So the overall average is closer to two but it was still surprisingly low considering the fact 64 men and women participate. These results are presented below. Again we used a similar traffic pattern of bursty and Bernoulli arrivals. Though the average number of iterations increases slowly with N fewer than $O(\log N)$ iterations are required most of the times.

Iterations	Frequency
1	13598
2	34262
3	2111
4	29

Table 2: Iterations for 64 x 64switch

These low numbers seemed to indicate the average sizes of the preference lists must be rather low. We investigated this value. We observed that the input preference lists have

pretty low average lengths. For example in the first 1000 iterations for a 32 x 32 switch we noticed that only in fewer than thirty schedulings were the input preference lists more than one and even then for the first iterations this average was close to two. Thus they would become ineligible to participate within one or two rounds. On the other had the output preference lists grow to sizes of 5, 6 frequently. This seemed to indicate that the low convergence time was due to the fact the number of cells of slackness zero at a particular input was rather low. It is not known yet whether we can bound the number of cells of slackness zero at a particular input.

We noticed that the simulation slows down tremendously for a 64 size switch. This probably is due to the overhead of determining the slackness and output cushion of the first cells in the VOQs. For a 64 x 64 switch the simulation over 100,000 slots took around 2 hours to complete. However the 32 size switch took only about 15 minutes to complete. The implementation of the algorithm in the algorithm might prove to be difficult due to the need for computing per cell values. It would be interesting to see what the effect of limiting the number of iterations to a fixed maximum. A few cells would have their times of departure violated but this might be acceptable if most of the cells got through.

7 Conclusion

We have looked at the usage of the stable marriage algorithm for crossbar scheduling in combined input output queued switches. Approximate estimates for the average case performance of the general problem are much higher than the observed values for the CIOQ switch. For reasonable sized switches of 32 or 64 ports, the average number of iterations is close to two. This number increases slowly with the switch size. The reason behind this phenomenon seems to be the fact the the average input preferences are very small. From a practical viewpoint the drawback seems to be the computation of slackness for many cells. It remains to be seen whether the number of cells that participate in a scheduling phase can be bounded.

References

[AOST93] T. E. Anderson, S. Owicki, J.B. Saxe, and C. Thacker. High speed switch scheduling for local area networks. *ACM Transactions On Computer Systems*, 11(4):319–352, 1993.

[CGMP98] S. Chuang, A. Goel, N. McKeown, and B. Prabhakar. Matching output queueing with combined input output queueing. *Technical Report, Stanford CSL-TR-98-758*, 1998.

[CPK94] C.Y. Chang, A.J. Paulraj, and T. Kailath. A broadband packet switch architectire with input and output queueing. *IEEE Globecom*, 1:448–452, 1994.

[GS62] D. Gale and L.S. Shapley. College admissions and the stability of marriage. *American Mathematical Monthly*, 69:9–15, 1962.

[Knu76] D. E. Knuth. Mariages stables. 1976.