Practical Scheduling Algorithms for Input-Queued Switches

Jinhui Li and Nirwan Ansari
Center for Communications and
Signal Processing Research
Department of Electrical and
Computer Engineering
New Jersey Institute of Technology
University Heights, Newark,
NJ 07102, U.S.A.

Abstract — **The input-queued (IQ) switching architecture is becoming an attractive alternative for high speed switches owing to its scalability. In this paper, two new algorithms, referred to as maximum credit first (MCF) algorithm and iterative maximum credit first (IMCF) algorithm, are introduced. Theoretic analysis shows that the credits of an IQ switch using MCF are bounded for all admissible rate reservations. Simulations show that both MCF and IMCF have similar performance as the Birkhoff-von Neumann algorithm which can provide cell delay bound and 100% throughput in terms of QoS guarantees, but have lower off-line computational and on-line memory complexity.**

I. Introduction

There are two basic types of switching architectures: output-queued (OQ) and input-queued (IQ) switching architecture. When a packet arrives at an OQ switch, it is queued in its output queue immediately. The packet will stay in the output queue until it is transmitted from the switch, and hence 100% throughput can be achieved in OQ switches. OQ switches can also provide quality of service (QoS) guarantees by using scheduling mechanisms [1] such as WFQ[2]/PGPS[3] and WF^2Q [4]. The problem of OQ switches is that the fabric of an $N \times N$ OQ switch must run in the worst case N times as fast as its line rate, i.e., the speedup of an OQ switch is N. On the other hand, when a packet arrives at an IQ switch, it is placed in its input queue until it can be scheduled across the fabric. The packet can be transmitted out of the IQ switch immediately when it arrives the output port. The fabric of an IQ switch needs only run as fast as the line rate, i.e., the speedup of an IQ switch is 1.

In the high speed networks, fabric and memories with a bandwidth operated at N times the line rate may be infeasible. Thus, the IQ switching architecture has been adopted for high speed switch implementation owing to its scalability. One of the major problems with the IQ switching architecture is the head-of-line (HOL) blocking: the HOL cell, which cannot be forwarded because of output contention, can block the output-contention-free cells in the same queue when FIFO is used. HOL blocking limits the throughput of the IQ switch using a single FIFO queue in each input to approximately $2 - \sqrt{2} \approx 0.586$ even under *i.i.d.* Bernoulli traffic when N is large [5]. Under bursty traffic, it is even worse: the maximum throughput of such a switch decreases monotonically with the burstiness of traffic, and reaches to 0.5 when burstiness is large [6]. Stationary blocking is another problem for FIFO IQ switches: The total throughput of the switch can be as small as the throughput of a single link under certain periodic traffic, even when N is very large [7].

[1]This work was supported in part by Lucent Technologies, and the New Jersey Commission on Science and Technology via the New Jersey Center for Wireless Telecommunications.

Previous research [8][9][10] shows that HOL blocking can be completely eliminated in IQ switches by adopting virtual output queueing (VOQ), in which multiple VOQs directed to different outputs are maintained at each input. Also, the throughput of an IQ switch can be increased to 100% under all admissible independent traffic by using well designed scheduling algorithms such as longest queue first (LQF) [9] algorithm and oldest cell first (OCF) [10] algorithm. Though 100% throughput can be achieved using IQ switches, other QoS features such as bandwidth and cell delay still cannot be guaranteed using the above algorithms.

Another approach to reduce HOL blocking is to increase the speed of the fabric. The ratio of bandwidth between fabric and input link is defined as speedup. When speedup is larger than 1 and smaller than N, buffers are required at the outputs as well as inputs. This switching architecture with both input and output buffering is called combined input output queueing (CIOQ) switch.

Enormous efforts have been made on providing QoS guarantees with CIOQ switch. Recently, Chuang *et al.* proved [11] that a CIOQ switch using stable matching [12] algorithm and critical cells first (CCF) insertion policy with speed up equal to two can exactly mimic an OQ switch that uses push-in first-out (PIFO) queueing policy.

Other efforts have been made to achieve QoS guarantees by IQ switch without speedup. Schedule tables in [8] and Store-sort-and-forward (SSF) algorithm [13] can guarantee cell delay with a fixed schedule that is pre-computed when connections are setup. However, they have problems such as computational complexity and rate granularity limitation.

Chang *et al.* proposed [14] the Birkhoff-von Neumann algorithm based on a decomposition result by Birkhoff and von Neumann for a doubly substochastic matrix. This algorithm can provide 100% throughput for all non-uniform traffic. Furthermore, if the traffic is (σ, r)-upper constrained, cell delay can be deterministically guaranteed using this algorithm. The problem of this algorithm is that the off-line computational complexity is too high: $O(N^{4.5})$. If the assigned rate of the sessions change frequently, which is more likely in a large switch, the algorithm will be impractical. In this paper, we proposed two new algorithms, maximum credit first (MCF) algorithm and iterative maximum credit first (IMCF) algorithm, which have lower off-line complexity and similar performance as the Birkhoff-von Neumann algorithm, and more realizable.

The rest of the paper is organized as follows. In Section 2, we describe our switch model and algorithms. Discussion and simulation results of the proposed algorithms are presented in Section 3. Concluding remarks are given in Section 4.

II. Our Switch Model and Algorithms

Our $N \times N$ input-queued switch which has no speedup consisting of N inputs, N outputs and a non-blocking switch fabric such as crossbar. The packets, which may have variable length, are broken into fixed length cells when they are arriving in the inputs. After the cells crossed the fabric, they are reassembled to the original variable length packets. Time slot is defined as the time required to transmit a cell with the line rate. Virtual output queueing is adopted in order to eliminate the HOL blocking. We denote the VOQ directed to output j at input i as $Q_{i,j}$.

The basic objective of scheduling an input-queued switch is to find a contention free matching which is equivalent to solving a bipartite graph matching problem, as shown in Fig. 1(a). Each vertex on the left side represents an input, and that on the right side represents an output. There have an edge between every input vertex i and every output vertex j. Associated with each edge is a weight, $w_{i,j}$, which is defined differently by different algorithms. In Fig. 1(a), the edges with the weight of 0 are omitted. The scheduler selects a matching between the inputs and outputs with the constraints of unique pairing, i.e., at most one input can be matched to each output, and vice versa. Then, a cell is transmitted per matched input-output pair if there has a cell in $Q_{i,j}$. A maximum weighted matching algorithm computes a matching which can maximize the aggregate weight. Fig. 1(b) is the maximum weighted matching solution of Fig. 1(a).

Suppose the rate assigned to the traffic from input i to output j is $r_{i,j}$, which is also the arrival rate of $Q_{i,j}$. The traffic is admissible if and only if the following inequalities are satisfied:

$$\sum_{j=1}^{N} r_{i,j} \leq 1, \forall i, \tag{1}$$

$$\sum_{i=1}^{N} r_{i,j} \leq 1, \forall j. \tag{2}$$

$R = (r_{i,j})$ is a doubly substochastic matrix. For any doubly substochastic matrix R, there exists [14][15] a doubly stochastic matrix $\tilde{R} = (\tilde{r}_{i,j})$ such that $r_{i,j} \leq \tilde{r}_{i,j}, \forall i, j$. Matrix $\tilde{R}$ is a doubly stochastic matrix, if it satisfies

$$\sum_{j=i}^{N} \tilde{r}_{i,j} = 1, \forall i, \tag{3}$$

and

$$\sum_{i=i}^{N} \tilde{r}_{i,j} = 1, \forall j. \tag{4}$$

$\tilde{R}$ can be constructed by the following algorithm.

Algorithm 1

1. *Define* $ra_i = 1 - \sum_{j=1}^{j=N} r_{i,j}$. *Calculate* ra_i *for all* i.
2. *Define* $rb_j = 1 - \sum_{i=1}^{i=N} r_{i,j}$. *Calculate* rb_j *for all* j.
3. *Calculate* $rr = N - \sum_{i=1,j=1}^{i=N,j=N} ri, j$.
4. *Let* $\tilde{r}_{i,j} = r_{i,j} + \frac{ra_i rb_j}{rr}$.

The computational complexity of Algorithm 1 is $O(N^2)$.

Suppose every $Q_{i,j}$ has a credit $c_{i,j}$ which is a real variable that has the initial value of 0. At the beginning of every time slot, $c_{i,j}$ increases by $\tilde{r}_{i,j}$. Then the scheduler selects the matching according to the current credits of VOQs. Define service matrix at time slot n as $S(n) = (s_{i,j}(n))$, where $s_{i,j}(n)$ equals to 1 if input-output pair (i,j) is in the matching, otherwise $s_{i,j}(n) = 0$. In this paper, we only consider permutation matrices as the service matrix S, implying that there are always exactly N pairs in every matching. From all the permutation matrices, the MCF algorithm selects the one which can maximize the aggregate credit, i.e.,

$$\arg\max_{S}[\sum_{i,j} s_{i,j}(n)c_{i,j}(n)],$$

where, $\sum_j s_{i,j}(n) = \sum_i s_{i,j}(n) = 1, \forall i, j$. The service matrix S can be found by maximum weighted matching algorithm which has a computational complexity of $O(N^3)$ [16]. The cells in VOQs are transmitted across the fabric according to their corresponding values in service matrix S: If $s_{i,j}$ equals to 1 and $Q_{i,j}$ is not empty, then the HOL cell of $Q_{i,j}$ will be transmitted to output j. In the meanwhile, $c_{i,j}$ decreases by $s_{i,j}$ for all i, j. Thus, $c_{i,j}(n)$, the credit of $Q_{i,j}$ at the end of time slot n, is

$$c_{i,j}(n) = c_{i,j}(n-1) - s_{i,j}(n) + \tilde{r}_{i,j}. \tag{5}$$

where, $c_{i,j}(n-1)$ is the credit of $Q_{i,j}$ at the end of time slot $n-1$.

Owing to the $O(N^3)$ complexity, MCF algorithm is difficult to implement in high speed networks, so we propose an iterative approximation of MCF algorithm: iterative maximum credit first (IMCF) algorithm. IMCF performs the following three steps in each iteration:

1. Request: Each unmatched input sends a request to every output.
2. Grant: If an unmatched output receives any requests, it chooses the one with the largest credit. Ties are broken randomly.
3. Accept: If an input receives any grants, it chooses the one with the largest credit. Ties are broken randomly.

IMCF stops when there has no unmatched input and output. It converges in at most N iterations, and the service matrix S selected by IMCF is always a permutation matrix.

Property 1 *At the end of any time slot, the credits of an IQ switch using MCF or IMCF satisfy the following equations:*

$$\left\{ \begin{array}{ll} \sum_{j=1}^{N} c_{i,j} = 0, & \forall i \\ \sum_{i=1}^{N} c_{i,j} = 0, & \forall j \\ \sum_{i,j} c_{i,j} = 0 & \end{array} \right. . \tag{6}$$

Property 2 *In any time slot just before the service matrix is calculated, the credits of an IQ switch using MCF or IMCF satisfy the following equations:*

$$\left\{ \begin{array}{ll} \sum_{j=1}^{N} c_{i,j} = 1, & \forall i \\ \sum_{i=1}^{N} c_{i,j} = 1, & \forall j \\ \sum_{i,j} c_{i,j} = N & \end{array} \right. . \tag{7}$$

Theorem 1 *The credits of an IQ switch using MCF algorithm are bounded for all admissible rate reservations, i.e.,* $|c_{i,j}| < \infty$.

III. Discussion and Simulations

Let $S_{i,j}(n)$ be the cumulative number of slots that are assigned to $Q_{i,j}$ by time slot n. Then the credit of $Q_{i,j}$ is

$$c_{i,j}(n) = \tilde{r}_{i,j} \cdot n - S_{i,j}(n). \quad (8)$$

For any $n \geq m$,

$$S_{i,j}(n) - S_{i,j}(m) = \tilde{r}_{i,j}(n-m) - c_{i,j}(n) + c_{i,j}(m). \quad (9)$$

If we assume the lower bound and upper bound of the credit are c^- and c^+, respectively, and let $\Delta_c = c^+ - c^-$, then

$$r_{i,j}(n-m) - \Delta_c \leq S_{i,j}(n) - S_{i,j}(m) \leq r_{i,j}(n-m) + \Delta_c. \quad (10)$$

Eq. (10) implies that if $A_{i,j}$, the traffic from input i to output j, conforms to $(\sigma_{i,j}, r_{i,j})$, i.e.,

$$A_{i,j}(n) - A_{i,j}(m) \leq r_{i,j}(n-m) + \sigma_{i,j}, \quad (11)$$

then the cell delay from input i to output j is bounded by $\lceil(\sigma_{i,j} + \Delta_c)/r_{i,j}\rceil$.

Chang *et al.* show [14] if Eq. (1) and (2) are satisfied, then the Birkhoff-von Neumann algorithm can guarantee

$$S_{i,j}(n) - S_{i,j}(m) \geq r_{i,j}(n-m) - u_{i,j}, \quad (12)$$

for all i, j, $n \geq m$, where $u_{i,j} \leq U = N^2 - 2N + 2$.

Comparing Eq. (10) with Eq. (12), we can say if the Δ_c of MCF and IMCF is comparable with $U = N^2 - 2N + 2$, then MCF and IMCF are as good as the Birkhoff-von Neumann algorithm in terms of QoS guarantees. Simulation results of c^+, c^-, and Δ_c are shown in Table 1. The simulations are made under various non-uniform rate reservations on input-queued switches with $N = 16$, $N = 32$, and $N = 64$. Table 1 shows the Δ_c of MCF and IMCF, which are much smaller than U.

Fig. 2 shows the distribution of the percentage of cells which experience various delays over a 16×16 IQ switch using MCF, IMCF and the the Birkhoff-von Neumann algorithm under non-uniform traffic with a total traffic load of 90%. Traffic $A_{i,j}$ conforms to $(\sigma_{i,j}, r_{i,j})$ for all i, j, in which $\sigma_{i,j} = 1000 r_{i,j}$. Thus, the designed delay bound is $1000 + \lceil u_{i,j}/r_{i,j} \rceil$ time slots for the Birkhoff-von Neumann algorithm, and is $1000 + \lceil \Delta_c / r_{i,j} \rceil$ time slots for MCF and IMCF. The simulation result shows that the cell delay bound is less than 1200 time slots for all the algorithms as we expected. Fig. 2 demonstrates that the performance of the the Birkhoff-von Neumann algorithm and IMCF are almost identical, and MCF has a tighter bound than the other two algorithms.

When $r_{i,j}$, the reserved rate between input i and output j, changes, MCF and IMCF need to recalculate $\tilde{R}$. On the other hand, the Birkhoff-von Neumann algorithm not only need to recalculate $\tilde{R}$, but also need to decompose it again. It has the complexity of $O(N^{4.5})$, and is termed as off-line computational complexity in [14]. Owing to the high off-line computational complexity, the Birkhoff-von Neumann algorithm will be hard to implement in a dynamic environment. Table 2 compares the computational and memory complexity of the Birkhoff-von Neumann algorithm, MCF, and IMCF. It indicates that MCF and IMCF have higher on-line computational complexity, but have lower off-line computational complexity and on-line memory complexity.

IV. Conclusions

In this paper, we proposed two new algorithms, MCF and IMCF, which have the similar performance as the Birkhoff-von Neumann algorithm in terms of QoS guarantees, but have less off-line computational and on-line memory complexity, which makes these new algorithms easier to be implemented in practice.

References

[1] H. Zhang, "Service disciplines for guaranteed performance service in packet-switching networks," *Proc. IEEE,* 83(10), pp. 1374 -1396, 1995.

[2] A. Demers, S. Keshav, and S. Shenkar, "Analysis and simulation of a fair queueing algorithm," *Internet. Res. and Exper.,* vol. 1, 1990.

[3] A.K. Parekh, R.G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: the single node case," *IEEE Transactions on Networking,* Vol. 1, No. 3, pp. 344-357, June 1993.

[4] J.C.R. Bennett and H. Zhang, "WF^2Q: worst-case fair weighted fair queueing", *Proc. INFOCOM'96,* March 1996, San Francisco, CA.

[5] M. Karol, M. Hluchyi, S. Mogan, "Input versus output queueing on a space-division packet switch," *IEEE Trans. Commun.,* vol. COM-35, pp. 1347-1356, Dec. 1987.

[6] S.-Q. Li, "Performance of a nonblocking space-division packet switch with correlated input traffic," *Proc. IEEE Globecom,* pp. 1754-1763, 1989.

[7] S.-Y. Li, "Theory of periodic contention and its application to packet switching," *Proc. INFOCOM'88,* pp. 320-325, March 1988.

[8] T. Anderson, S. Owicki, J. Saxe and C. Thacker, "High speed switch scheduling for local area networks," *ACM Trans. on Computer Systems,* pp. 319-352, Nov. 1993.

[9] N. Mckeown, V. Anantharam and J. Walrand, "Achieving 100% throughput in an input-queued switch," *Proc. INFOCOM'96,* pp. 296-302, San Francisco, CA, March 1996.

[10] A. Mekkittikul and N. McKeown, "A starvation-free algorithm for achieving 100% throughput in an input-queued switch," *Proc. ICCCN'96*, pp. 226-231, Oct. 1996.

[11] S.-T. Chuang, A. Goel, N. Nckeown, and B. Prabhakar, "Matching output queueing with a combined input/output-queued switch," *IEEE J. Select. Areas Commun.,* vol. 17, No. 6, pp. 1030-1039, June 1999.

[12] D. Gale and L.S. Shapley, "College admissions and the stability of marriage," *American Mathematical Monthly,* Vol. 69, pp9-15, 1962.

[13] S. Li and N. Ansari, "Input queued switching with QoS guarantees," *Proc. IEEE INFOCOM,* pp. 1152-1159, Mar. 1999.

[14] C.-S. Chang, W.-J. Chen, H.-Y. Huang, "On service guarantees for input buffered crossbar switches: a capacity decomposition approach by Birkhoff and von Neumann," *IWQoS'99,*, pp. 79-86, 1999.

[15] J. von Neumann, "A certain zero-sum two-person game equivalent to the optimal assignment problem," *Contributions to the Theory of Games,* Vol. 2, pp. 5-12, Princeton University Press, Princeton, NJ, 1953.

[16] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin, *Network Flows: Theory, Algorithms, and Applications,* Prentice Hall, Englewood Cliffs NJ, 1993.

TABLE 1. Maximum and minimum credit

N	16			32			64		
U	226			962			3970		
Algorithm	c^+	c^-	Δ_c	c^+	c^-	Δ_c	c^+	c^-	Δ_c
MCF	1.08	-1.07	2.15	1.05	-0.96	2.01	0.95	-0.98	1.93
IMCF	1.36	-3.69	5.05	1.06	-4.15	5.21	1.16	-3.35	4.51

TABLE 2. Computational and memory complexity

Algorithm	off-line complexity	on-line complexity	on-line memory complexity
Birkhoff-von Neumann	$O(N^{4.5})$	$O(logN)$	$O(N^3 logN)$
MCF	$O(N^2)$	$O(N^3)$	$O(N^2)$
IMCF	$O(N^2)$	$O(N^2)$	$O(N^2)$

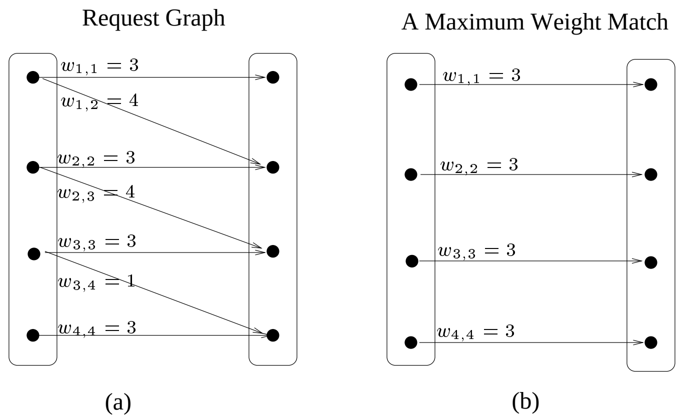

Figure 1: A Bipartite graph matching example: (a) the request graph, (b) a maximum weighted matching.

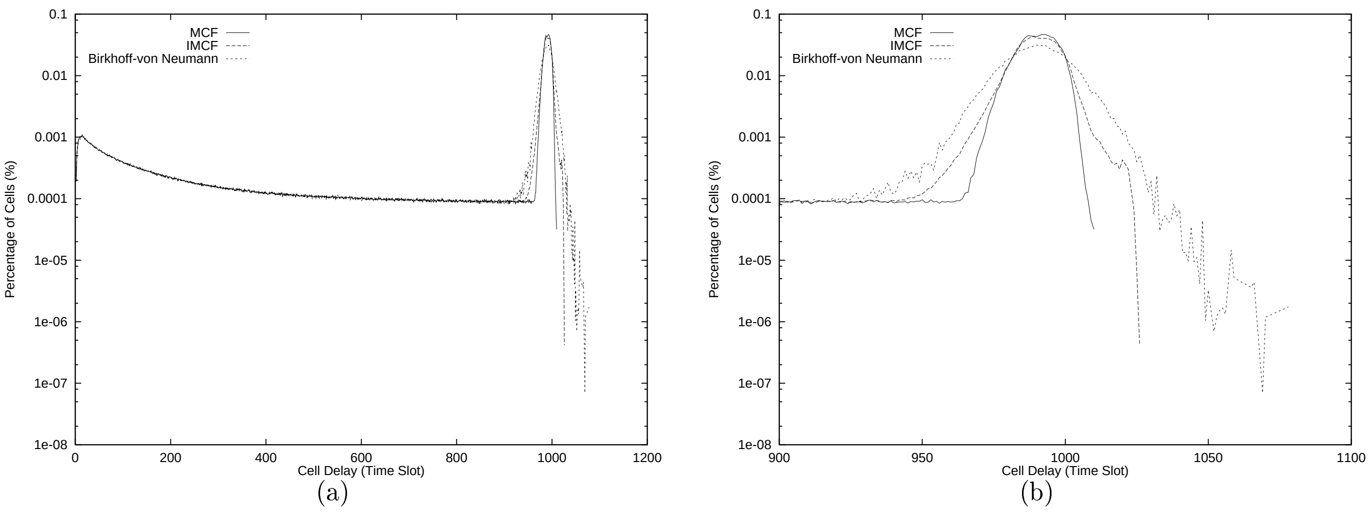

Figure 2: Cell delay distribution