PAPER

Efficient Scheduling for SDMG CIOQ Switches

Mei YANG† ***and*** **S.Q. ZHENG**††, ***Nonmembers***

SUMMARY Combined input and output queuing (CIOQ) switches are being considered as high-performance switch architectures due to their ability to achieve 100% throughput and perfectly emulate output queuing (OQ) switch performance with a small speedup factor S. To realize a speedup factor S, a conventional CIOQ switch requires the switching fabric and memories to operate S times faster than the line rate. In this paper, we propose to use a CIOQ switch with space-division multiplexing expansion and grouped input/output ports (SDMG CIOQ switch for short) to realize speedup while only requiring the switching fabric and memories to operate at the line rate. The cell scheduling problem for the SDMG CIOQ switch is abstracted as a bipartite k-matching problem. Using fluid model techniques, we prove that any maximal size k-matching algorithm on an SDMG CIOQ switch with an expansion factor 2 can achieve 100% throughput assuming input line arrivals satisfy the strong law of large numbers (SLLN) and no input/output line is oversubscribed. We further propose an efficient and starvation-free maximal size k-matching scheduling algorithm, kFRR, for the SDMG CIOQ switch. Simulation results show that kFRR achieves 100% throughput for SDMG CIOQ switches with an expansion factor 2 under two SLLN traffic models, uniform traffic and polarized traffic, confirming our analysis.

key words: *CIOQ switch, cell scheduling, maximal size matching, speedup.*

1. Introduction*

Output queuing (OQ) switches are employed for many commercial switching systems today due to their ability to maximize throughput and provide quality of service (QoS) guarantees. However, OQ switches are not scalable for high line rates and/or large numbers of ports since the switching fabric and memories for an $N \times N$ OQ switch are required to run N times faster than the line rate. On the other hand, input queuing (IQ) switches are scalable with their switching fabric and memories operating at the line rate, but IQ switches have a limited throughput because of head-of-line (HOL) blocking and cannot provide QoS guarantees. To reduce the speed requirement of the switching fabric and memories of OQ switches and improve the switch performance of IQ switches, combined input and output queuing (CIOQ) switches are proposed. CIOQ switches are being considered as high-performance switch architectures due to their ability to achieve 100% throughput and even emulate OQ switch performance with a small speedup factor [2]. Fig. 1 shows an $N \times N$ CIOQ switch. To remove head-of-line (HOL) blocking [13], each input port I_i maintains N virtual output queues (VOQs) with $Q_{i,j}$ buffering packets destined for output port O_j. With an internal speedup larger than 1, packets need to be buffered at outputs as well.

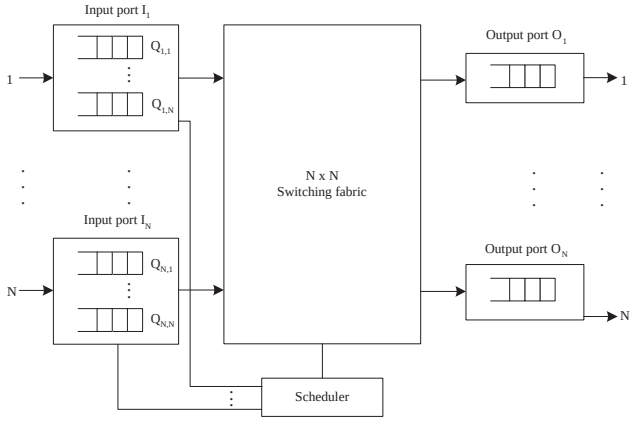

Fig. 1 An $N \times N$ CIOQ switch.

In this paper, we assume that all switches we discuss are cell based. In such a switch, variable-length packets are segmented into fixed-size cells upon arrival, transferred through the switching fabric, and reassembled back into original packets before they depart the switch. Time is divided into cell slots and one cell slot equals to the transmission time of a cell on the input/output line. In each cell slot, a scheduling algorithm selects a matching between input ports and output ports such that no input (resp. output) port may be matched to more than one output (resp. input) port. Fixed-size cells and slotted time switching make it easier for the scheduler to configure the switching fabric for high throughput [16].

The cell scheduling problem for VOQ based switches can be modelled as a bipartite matching problem [16]. Although maximum weight matching algorithms are proved to achieve 100% throughput for all admissible identically

Manuscript received May 20, 2005.
Manuscript revised December 3, 2005.
Final manuscript received December 3, 2005.

†The author is with the Department of Electrical and Computer Engineering, University of Nevada, Las Vegas, Las Vegas, NV 89154 USA (email: meiyang@egr.unlv.edu).

††The author is with the Department of Computer Science, University of Texas at Dallas, Richardson, TX 75080 USA (email: sizheng@utdallas.edu).

†Part of the results has been presented at IEEE INFOCOM 2003 [28].

independently distributed (i.i.d.) arrivals [17], they are infeasible for high speed implementation with their time complexity of $O(N^3 \log N)$ [25]. The most efficient maximum size matching algorithm has a time complexity of $O(N^{2.5})$ [11], [25]. However, maximum size matching algorithms are too complex for hardware implementation and can cause unfairness [17]. Most practical scheduling algorithms proposed, such as parallel iterative matching (PIM) [1], *i*SLIP [16], dual round-robin matching (DRRM) [5], first come first serve in round-robin matching (FIRM) [22], static round-robin (SRR) [12], iterative ping-pong arbitration (PPA) [4] scheme, and the round-robin priority matching (RRPM) [14], are iterative algorithms that find a maximal size matching to approximate a maximum size matching.

A switch with a speedup factor S can remove up to S cells from each input port and deliver up to S cells to each output port within one cell slot. Hence, an IQ switch has a speedup of 1, an OQ switch has a speedup of N, and a CIOQ switch has a speedup between 1 and N. It has been shown that a CIOQ switch with a speedup of 4 or 2 can exactly emulate an OQ switch by employing stable matching [9] based algorithms, such as the most urgent cell first algorithm (MUCFA) [21], the critical cell first (CCF) algorithm [6], and the just preferred matching (JPM) algorithm [24]. Unfortunately, these scheduling algorithms are highly impractical due to their high time complexity ($O(N^2)$ iterations).

In [8], Dai and Prabhakar proved that employing any maximal size matching algorithm a CIOQ switch with $S = 2$ can achieve 100% throughput for arbitrarily distributed input patterns such that input arrivals satisfy the strong law of large numbers (SLLN) and no input/output is oversubscribed. Since almost all real traffic processes satisfy these properties, this result has high practical significance for at least two reasons. First, achieving 100% throughput is a necessary condition for a CIOQ switch to realize OQ-equivalent quality of service (QoS) guarantees with carefully designed queuing disciplines at each VOQ and at each output queue. Second, maximal size matching algorithms are easier to implement than maximum size matching algorithms or stable matching algorithms. In addition, it is shown that CIOQ switches with any maximal size matching algorithms perform as good as OQ switches in terms of delay under Bernoulli i.i.d. arrival traffic [23].

To realize speedup for CIOQ switches, in the conventional scheme, it requires the switching fabric and memories to run S times faster than the line rate. Under current technology, the switching fabric can support up to 3.6Gbps line rate [26]. On the other hand, advances in fiber-optic transmission technologies have greatly pushed the increase of optical transmission rate. Each individual channel now can operate at OC-192 (10Gbps) or even OC-768 (40Gbps). Although silicon technologies have advanced rapidly, the gap between the data rate that optical transmission technology can deliver and the switching speed that electronic switching fabric can provide is becoming wider and wider [18]. Thus it may not always be feasible to run the switching fabric much faster than the line rate. Memories with sufficient access rate are simply not available for high line rate due to the limitation of current semiconductor technology. Even with fast switching fabric and memories, it may not be possible to run the cell scheduling algorithm fast enough to realize switch speedup greater than 1. In [27], we proposed pipelined maximal size matching algorithms to relax the running time for the scheduling algorithm for CIOQ switches with speedup. However, the running time for the switching fabric and memories is not relaxed.

To relax the stringent timing requirement of the operation speed of the switching fabric and memories, we introduce a CIOQ switch architecture with space-division multiplexing expansion and grouped input/output ports, shortened as an SDMG CIOQ switch. In an SDMG CIOQ switch, the number of connections between each input/output port and the switching fabric is increased, but the switching fabric only needs to run as fast as the line rate. We define the expansion factor of an SDMG CIOQ switch as the ratio of the number of connections between an input/output port and the switching fabric and the number of input/output lines associated with an input/output port.

We model the cell scheduling problem for SDMG CIOQ switches as a bipartite k-matching problem. Using fluid model techniques, we prove that any maximal size k-matching algorithm for an SDMG CIOQ switch with an expansion factor 2 can achieve 100% throughput assuming that input line traffic arrivals satisfy SLLN and no input/output line is oversubscribed. We propose the k-connection FIRM-based round-robin (kFRR) algorithm to find maximal size k-matchings on SDMG CIOQ switches. Through simulations, we show that the kFRR algorithm achieves 100% throughput for SDMG CIOQ switches with an expansion factor 2 under two SLLN traffic models: uniform traffic (both Bernoulli arrivals and bursty arrivals) and polarized traffic. This confirms our analysis based on fluid model techniques. The advantage of the proposed scheme compared to existing schemes is that it achieves the same performance as switches with speedup but only requires the switching fabric and memories to operate at the same speed as the line rate.

The remainder of this paper is organized as follows. Section 2 presents the SDMG CIOQ switch architecture. Section 3 defines and models the cell scheduling problem for SDMG CIOQ switches. Section 4 gives an analysis of the expansion factor that is sufficient for an SDMG CIOQ switch to achieve 100% throughput. Section 5 describes the kFRR scheduling algorithm and discusses its properties. Section 6 presents the simulation results of kFRR. Section 7 discusses possible hardware implementation schemes for the kFRR algorithm. Section 8 concludes the paper.

2. SDMG CIOQ Switches*

We assume all the switch architectures we discuss are cell based. To realize the speedup required for a CIOQ switch, we consider an alternative CIOQ switch architecture with more connections between each input/output port and the switching fabric. We generalize this CIOQ switch architecture by grouping multiple input/output lines into one port. The purpose of introducing grouped input/output ports is to achieve better buffer utilization [20], improve scheduling performance [19], and balance switch input/output loads. We name such a CIOQ switch as a *CIOQ switch with space-division multiplexing expansion and grouped input/output ports* (SDMG CIOQ switch for short). Fig. 2 shows an $N \times N$ SDMG CIOQ switch, where N is the number of input/output lines.

The characteristics of the SDMG CIOQ switch are

listed as follows.

- It has N/g (grouped) input ports denoted as I_i's, and N/g (grouped) output ports denoted as O_j's, where $1 \le i, j \le N/g$. Input port I_i groups input lines L_l's, $(i-1)g+1 \le l \le ig$, and output port O_j groups output lines M_m's, $(j-1)g+1 \le m \le jg$. g is called the *group factor*, $1 \le g \le N$. In practice, g is selected appropriately to balance the performance and implementation complexity.
- Each input port I_i maintains N/g VOQs with $Q_{i,j}$ buffering cells destined for output port O_j, $1 \le i, j \le N/g$.
- Each output port O_j maintains g output queues, each associated with an output line.
- It has an $Nk/g \times Nk/g$ switching fabric with k connections to each input/output port. We assume that the switching fabric is non-blocking or rearrangeable non-blocking. k is called the *port connection factor*, and it is assumed that $k \ge g$.
- Cells belonging to one VOQ of an input port may be transmitted through the switching fabric simultaneously. The sequence of cells can be kept by appropriately setting the switching fabric such that the cell order is consistent with the connection order.
- A cell in an input port can be switched to its destination output port through any of the k connections between the input port and the switching fabric and any of the k connections between the switching fabric and the destination output port.

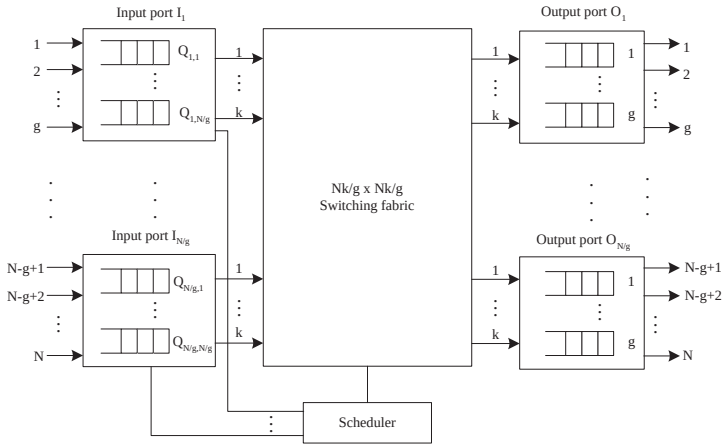

Fig. 2 An SDMG CIOQ switch.

We define $P = k/g$ as the *expansion factor* of an SDMG CIOQ switch. To relax the memory access rate, the interface between each VOQ (or output queue) and the switching fabric is expanded to multiple copies to allow more than one cell to be transmitted from a VOQ (or into an output queue). Fig. 3 (a) shows a possible queuing scheme at an input port, in which each VOQ is composed of k sub-queues. Cells belonging to one VOQ are buffered in sub-queues following the order of $1, 2, \cdots, k$. Since $k \ge g$, it is feasible for each VOQ to receive up to g cells (one to each sub-queue) coming from different input lines without speedup. A queue controller (QC) is used to select up to g out of k sub-queues to receive these incoming cells. Since up to k cells (in different sub-queues) from one VOQ may be sent to the switching fabric through up to k connections in one cell slot, an interconnection controller (IC) is used as the interface between each VOQ and the k connections to ensure the cells are sent in the same sequence as they arrive. Fig. 3 (b) shows a possible queuing scheme at an output port, where each output queue is composed of k sub-queues. An IC is used as the interface between k connections and each output queue to ensure the cells enter the sub-queues in the same sequence as they are transmitted on the k connections. Each output queue is connected to its corresponding output line. In this scheme, since in one cell slot at most one cell enters or leaves a sub-queue, memory speedup is not needed.

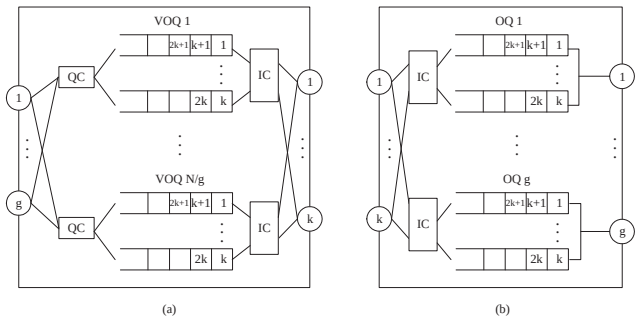

Fig. 3 (a) Queue structure at the input port. (b) Queue structure at the output port.

In [19], Obara *et al.* proposed a similar switch architecture to enhance the scheduling performance for an ATM switch. Our purpose of using the SDMG CIOQ switch architecture is to achieve speedup but only require the switching fabric and memories to operate at the same speed as the line rate. The tradeoff of the SDMG CIOQ switch is the increased complexity of the switching fabric and the added QCs and ICs in input/output ports. With current semiconductor technology, it is feasible to implement the SDMG CIOQ switch with regular size g and k (for the switch sizes discussed in Section 6).

3. Cell Scheduling for SDMG CIOQ Switches

For an SDMG CIOQ switch, the scheduling algorithm needs to determine a conflict-free switching fabric setting for switching cells from input ports to output ports in each cell slot. The cell scheduling problem for the SDMG CIOQ switch can be modelled as a bipartite k-matching problem on the graph $G = (V, E)$, where $V = V_1 \cup V_2$, $V_1 = \{\text{input ports}\}$, $V_2 = \{\text{output ports}\}$, $\mid V_1 \mid = \mid V_2 \mid = N/g$, $E = \{\text{connection requests from input ports to output ports}\}$. Let $M = \mid E \mid$. In each cell slot, there might be up to k connection requests from an input port to an output port. Therefore, G may not be a simple graph since there may be more than one edge between one pair of nodes.

A *k-matching* is an edge set $\mathcal{K} \subseteq E$ such that no node of G is incident by more than k edges in $\mathcal{K}$, where $k \ge 1$. A *matching* is a special case of k-matching with $k = 1$. A *match* is an edge $(i, j) \in \mathcal{K}$. A *perfect k-matching* $\mathcal{K}$ is one

that each node of G is incident by k edges in $\mathcal{K}$. A *maximum size k-matching* is one with the maximum number of edges. A *maximal size k-matching* is one that is not contained in any other k-matchings.

Fact 1: For a maximal size k-matching of $\mathcal{K} \subseteq G$, all the following statements are true. (1) The number of matches between any I_i and any O_j in $\mathcal{K}$ is less than or equal to k. (2) The total number of matches between any I_i and all O_j's in $\mathcal{K}$ is less than or equal to k. (3) The number of matches between all I_i's and any O_j in $\mathcal{K}$ is less than or equal to k. (4) If there are at least k connection requests between some I_i and some O_j, then at least one of the following holds: (a) I_i has k matches to some output ports, and (b) O_j has k matches to some input ports.

Fig. 4 compares a maximum size 2-matching and a maximal size 2-matching for a 4×4 SDMG CIOQ switch with $g = 1$ and $k = 2$. With the maximum size 2-matching shown in Fig. 4(b), $Q_{1,1}$, $Q_{1,3}$, $Q_{2,2}$, $Q_{2,4}$, and $Q_{3,2}$ will be served.

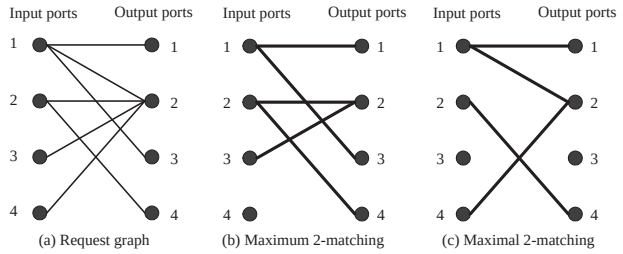

Fig. 4 A maximum size 2-matching and a maximal size 2-matching of a 4×4 SDMG CIOQ switch.

As a special case of the bipartite b-matching problem [7], the maximum bipartite k-matching problem can be transformed to a maximum-flow problem in $O(M)$ time. Since the transformed flow network is a unit network [25], we can use Dinic's algorithm to solve the corresponding maximum-flow problem in $O(\sqrt{N}M)$ time [25]. However, this algorithm is too complex for hardware implementation. Another noticeable problem with maximum size k-matching algorithms is that they may cause unfairness. For example, in Fig. 4, if $Q_{1,1}$, $Q_{1,3}$, $Q_{2,2}$, $Q_{2,4}$, and $Q_{3,2}$ continue having requests and other VOQs continue having no request in successive cell slots, then $Q_{1,2}$ may get starved since edge $(1, 2)$ does not belong to any maximum size 2-matching.

For practical use, we desire scheduling algorithms to be fast, starvation-free, easy to implement, and of high throughput [16]. Compared with maximum size k-matching algorithms, maximal size k-matching algorithms are simpler and possible to avoid unfairness. However, how good the performance of maximal size k-matching algorithms can be? In the following, we will give an analysis based on fluid model techniques.

4. Analysis of Maximal Size k-Matching Algorithms

We follow the definitions of SLLN and fluid model used in [8]. We define $A'_{l,m}(n)$ as the cumulative number of cells that have arrived from input line L_l destined for output line M_m up to cell slot n. We assume that the arrival processes $\{A'_{l,m}(\cdot), l, m = 1, ..., N\}$ satisfy a strong law of large numbers (SLLN), i.e. with probability one,

$$\lim_{n\to\infty} \frac{A'_{l,m}(n)}{n} = \lambda'_{l,m}, l, m = 1, ..., N, \tag{1}$$

where $\lambda'_{l,m}$ is called the cell arrival rate from input line L_l to output line M_m. We also assume that no input/output line is oversubscribed, i.e.

$$\forall 1 \le l, m \le N, \sum_{m=1}^{N} \lambda'_{l,m} \le 1, \sum_{l=1}^{N} \lambda'_{l,m} \le 1. \tag{2}$$

Equations (1) and (2) are very mild conditions. Almost all real traffic processes satisfy these two equations. Let $D'_{l,m}$ be the departure rate of cells coming from input line L_l to output line M_m. An SDMG CIOQ switch under a k-matching algorithm is said to be *work conserving* if

$$\lim_{n\to\infty} \frac{\sum_l D'_{l,m}(n)}{n} = \sum_l \lambda'_{l,m} \tag{3}$$

for any (traffic) arrival satisfying Equations (1) and (2), i.e. the long-run fraction of time that output line M_m ($1 \le m \le N$) is busy is equal to the cell arrival rate at the output line. This is equivalent to saying that the SDMG CIOQ switch can achieve 100% throughput if there is enough offered load.

We define $A_{i,j}(n)$ as the cumulative number of cells arrived at $Q_{i,j}$ (i.e. cells arriving at input port I_i and destined for output port O_j) up to cell slot n. For arrival processes $A'_{l,m}(\cdot)$'s satisfying Equation (1), we derive that arrival processes $\{A_{i,j}(\cdot), i, j = 1, ..., N/g\}$ also satisfy SLLN since with probability one,

$$\begin{aligned} &\lim_{n\to\infty} \frac{A_{i,j}(n)}{n} \\ =& \lim_{n\to\infty} \frac{\sum_{m=(j-1)g+1}^{jg} \sum_{l=(i-1)g+1}^{ig} A'_{l,m}(n)}{n} \\ =& \sum_{m=(j-1)g+1}^{jg} \sum_{l=(i-1)g+1}^{ig} \lambda'_{l,m} \\ =& \lambda_{i,j},\ i, j = 1, ..., N/g. \end{aligned} \tag{4}$$

We call $\lambda_{i,j}$ the cell arrival rate at $Q_{i,j}$. For arrival processes $A'_{l,m}(\cdot)$'s satisfying Equation (2), no input/output port is oversubscribed since

$$\begin{aligned} &\forall 1 \le i, j \le N/g, \\ &\sum_{j=1}^{N/g} \lambda_{i,j} = \sum_{m=1}^{N} \sum_{l=(i-1)g+1}^{ig} \lambda'_{l,m} \le g, \\ &\sum_{i=1}^{N/g} \lambda_{i,j} = \sum_{l=1}^{N} \sum_{m=(j-1)g+1}^{jg} \lambda'_{l,m} \le g. \end{aligned} \tag{5}$$

Let $D_{i,j}(n)$ be the number of cells departing from $Q_{i,j}$ up to cell slot n. We say an SDMG CIOQ switch under a

matching algorithm is *VOQ rate stable* if with probability one,

$$\lim_{n\to\infty} \frac{D_{i,j}(n)}{n} = \lambda_{i,j},\ i,j = 1,...,N/g \tag{6}$$

for any arrival process satisfying Equation (4). And an SDMG CIOQ switch is said to be *port conserving* if Equations (5) and (6) hold, which means that the cell arrival rate at output port O_j satisfies

$$\lim_{n\to\infty} \frac{\sum_{i=1}^{N/g} D_{i,j}(n)}{n} \le g, 1 \le j \le N/g.$$

Let an $N/g \times N/g$ matrix $Z(n)$ be the request matrix at cell slot n, where $Z_{i,j}(n)$ denotes the number of cells in $Q_{i,j}$ at the beginning of cell slot n. A maximal size k-matching algorithm $\mathcal{A}$ determines a matrix $\pi(n)$ in cell slot n, where $\pi_{i,j}(n)$, $1 \le i,j \le N/g$, indicating how many cells can be transmitted from input port I_i to output port O_j during cell slot n. Since $\mathcal{A}$ is a maximal size k-matching algorithm, we have the following equations based on Fact 1.

$$\forall 1 \le i,j \le N/g, \pi(n)_{i,j} \le k, \tag{7}$$

$$\forall 1 \le i \le N/g, \sum_{j=1}^{N/g} \pi(n)_{i,j} \le k, \tag{8}$$

$$\forall 1 \le j \le N/g, \sum_{i=1}^{N/g} \pi(n)_{i,j} \le k, \tag{9}$$

$$\forall 1 \le i,j \le N/g, \sum_{j=1}^{N/g} \pi(n)_{i,j} + \sum_{i=1}^{N/g} \pi(n)_{i,j} \ge k, \quad \text{if } Z_{i,j}(n) \ge k. \tag{10}$$

We define $T_\pi^{\mathcal{A}}(n)$ as the cumulative amount of time that permutation π determined by $\mathcal{A}$ has been used by cell slot n. Assuming that Π is the set of matrices determined by $\mathcal{A}$ that satisfy Equations (7)-(10), the following equations hold for the SDMG CIOQ switch:

$$Z_{i,j}(n) = Z_{i,j}(0) + A_{i,j}(n) - D_{i,j}(n),$$

$$D_{i,j}(n) = \sum_{\pi\in\Pi} \pi_{i,j} T_\pi^{\mathcal{A}}(n),$$

$$\sum_{\pi\in\Pi} T_\pi^{\mathcal{A}}(n) = n.$$

where $n \ge 0$ and $i,j = 1,\cdots,N/g$.

Consider a deterministic, continuous fluid model of the SDMG CIOQ switch shown in Fig. 2 operating under the maximal size k-matching algorithm $\mathcal{A}$ with offered arrivals satisfying Equation (4). For each $t \ge 0$ and $i,j = 1,\cdots,N/g$, the fluid model is governed by the following set of equations:

$$Z_{i,j}(t) = Z_{i,j}(0) + \lambda_{i,j} t - D_{i,j}(t) \ge 0, \tag{11}$$

$$\dot{D}_{i,j}(t) = \sum_{\pi\in\Pi} \pi_{i,j} \dot{T}_\pi^{\mathcal{A}}(t) > 0, \tag{12}$$

$$T_\pi^{\mathcal{A}}(\cdot) \text{ is nondecreasing, and } \sum_{\pi\in\Pi} T_\pi^{\mathcal{A}}(t) = t, \tag{13}$$

where $\dot{f}$ denotes the derivative of function f at t, assuming f is differentiable at t. A solution to Equations (11)-(13) is said to be a fluid model solution. The fluid model of an SDMG CIOQ switch operating under a k-matching algorithm is said to be *VOQ weakly stable* if every fluid model solution (D,T,Z) has $Z(t) = 0$ for $t \ge 0$.

Theorem 1: An SDMG CIOQ switch under a k-matching algorithm is VOQ rate stable if its fluid model is VOQ weakly stable.

For detailed proof, please refer to the proof of Theorem 3 in [8]. We only give an intuitive explanation here. By Equation (11), we get $D_{i,j}(t) = \lambda_{i,j} t$ for $Z_{i,j}(t) = 0$, $t \ge 0$. Hence $\lim_{t\to\infty} \frac{D_{i,j}(t)}{t} = \lambda_{i,j}$.

Before we go further, we first state a simple lemma [8].

Lemma 1: Let $f : [0,\infty) \to [0,\infty)$ be an absolutely continuous function with $f(0) = 0$. Assume that $\dot{f}(t) \le 0$ for almost every t (wrt Lebesgue measure) such that $f(t) > 0$ and f is differentiable at t. Then $f(t) = 0$ for almost every $t \ge 0$.

Please refer to the proof of Lemma 1 in [8].

Let (D,T,Z) be a fluid model solution satisfying Equations (11)-(13) with $Z(0) = 0$. Let $R_i(t) = \sum_j Z_{i,j}(t)$ denote the total amount of fluid queued at input port I_i at time t and $S_j(t) = \sum_i Z_{i,j}(t)$ be the total amount of fluid destined for output port O_j and queued at some input ports at time t. Define $C_{i,j}(t) = R_i(t) + S_j(t)$. In addition to the fluid model Equations (11)-(13), we have the following lemma.

Lemma 2: For an SDMG CIOQ switch with expansion factor $P = k/g$ operating under a maximal size k-matching algorithm, each fluid limit must satisfy the following equation for $1 \le i,j \le N/g$:

$$\dot{C}_{i,j}(t) \le \sum_{j=1}^{N/g} \lambda_{i,j} + \sum_{i=1}^{N/g} \lambda_{i,j} - k, \quad \text{whenever } Z_{i,j}(t) > 0. \tag{14}$$

Proof is given in Appendix A.

Theorem 2: For an SDMG CIOQ switch shown in Fig. 2, any maximal size k-matching algorithm with $k = 2g$, i.e. $P = k/g = 2$, can achieve 100% throughput assuming that input line arrivals satisfy SLLN and no input/output line is oversubscribed.

Proof is given in Appendix B.

5. The *k*FRR Scheduling Algorithm*

In this section, we focus our study on practical maximal size k-matching algorithms, which can be developed based on iterative maximal size matching scheduling algorithms, such as PIM [1], *i*SLIP [16], DRRM [5], FIRM [22], SRR [12], and iterative PPA scheme [4]. Among these algorithms, round-robin based algorithms, such as *i*SLIP, DRRM, and FIRM, are more attractive than others because of their fairness and implementation simplicity. FIRM improves *i*SLIP by reducing the service guarantee time from $(N-1)^2+N^2$ cell slots to N^2 cell slots. It is starvation-free and easy to implement in hardware at high speed [22]. In the following, we generalize the idea of FIRM for the SDMG CIOQ switch and present the k-connection FIRM-based round-robin (*k*FRR) scheduling algorithm. Similar to FIRM, *k*FRR is also an iterative and round-robin based algorithm.

For input port I_i, let a_i, where $1 \le a_i \le N/g$, be its accept pointer indicating the accept starting position in the circular round-robin priority queue, and $C(I_i)$ be the number of available connections at I_i. For output port O_j, let g_j, where $1 \le g_j \le N/g$, be its grant pointer indicating the grant starting position in the circular round-robin priority queue, and $C(O_j)$ be the number of available connections at O_j. Prior to the first iteration of *k*FRR in any cell slot, we set $C(I_i) = C(O_j) = k$ for all $1 \le i, j \le N/g$.

In each cell slot, *k*FRR iteratively finds a k-matching. It terminates after a fixed number of iterations or after a non-profit iteration (i.e. a maximal size k-matching is found). Each iteration of *k*FRR consists of the following three steps.

Step 1: **Request**. $\forall I_i, 1 \le i \le N/g$, if I_i has any available connection and any unresolved request (an unresolved request is one to an output port with any available connection), it sends all unresolved requests to their corresponding O_j's.

Step 2: **Grant**. $\forall O_j, 1 \le j \le N/g$, if O_j has any available connection and receives any request, it grants $\min\{C(O_j)$, the number of requests to $O_j\}$ requests, starting from g_j. These grants are sent to their corresponding I_i's. g_j is updated to the first input port that receives O_j's grant but does not accept it in the Accept phase or the first input port that does not receive O_j's grant if all O_j's grants are accepted in the first iteration, starting from g_j in a circular manner (modulo N/g) if and only if in the *the first iteration*. $C(O_j)$ is updated to the number of available connections at O_j.

Step 3: **Accept**. $\forall I_i, 1 \le i \le N/g$, if I_i has any available connection and receives any grant, it accepts $\min\{C(I_i)$, the number of grants to $I_i\}$ grants starting from a_i. a_i is updated to the next position to the last output port whose grant is accepted by I_i in a circular manner (modulo N/g). $C(I_i)$ is updated to the number of available connections at I_i.

Fig. 5(a) shows how *k*FRR with one iteration works using an example for a 4×4 SDMG CIOQ switch with $k = 2$ under saturated load. Saturated load means at some cell slot, $\forall 1 \le i, j \le 4$, $Q_{i,j} > 0$, and input port traffic arrivals are maintained in such a manner that $Q_{i,j} > 0$ in the following cell slots. At the start of cell slot 0, we assume that $g_j = 1$ and $a_i = 1$ for all $1 \le i, j \le 4$. In the Request step, each input port I_i sends a request to each output port O_j, represented by an edge in the figure. In the Grant step, each O_j grants I_1 and I_2 since each O_j only has two connections available and $g_j = 1$ for all $1 \le j \le 4$. In the Accept step, both I_1 and I_2 accept grants from O_1 and O_2 since each of them only has two connections available and $a_1, a_2 = 1$. Finally a 2-matching of size 4 is found. a_1 and a_2 are updated to 3, while a_3 and a_4 are not updated; g_1 and g_2 are updated to 3, while g_3 and g_4 are not updated. Fig. 5(b) illustrates the desynchronization effect of grant pointers of *k*FRR with the previous example. After cell slot 0, due to the desynchronization of grant pointers, a perfect 2-matching is obtained at cell slot 1. For the same reason, perfect 2-matchings (with different patterns) are found in cell slots 2 and 3.

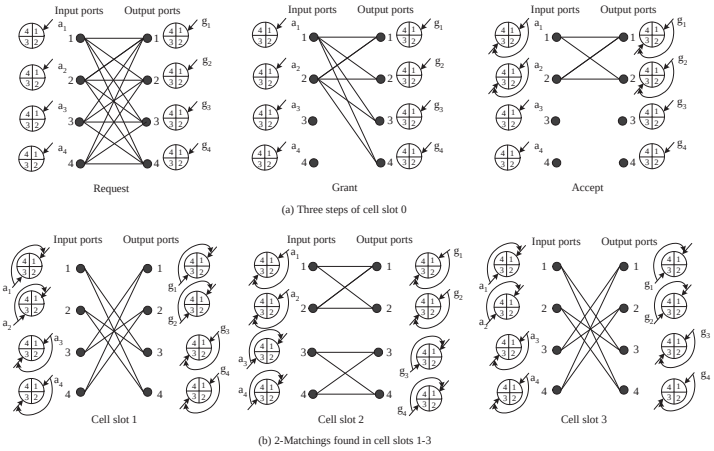

Fig. 5 An example of *k*FRR with one iteration.

*k*FRR has the following properties.

Property 1: At each output port O_j, due to the property of round-robin, the lowest priority input port is set as the one before the first input port that receives its grant but does not accept it in the first iteration or the input port before the first input port that does not receive O_j's grant if all O_j's grants are accepted in the first iteration.

Property 2: Under saturated load, all VOQs with a common output port have the same throughput because the grant pointer at the output port moves to each requesting input port in a fixed order (every $\frac{N}{kg}$ cell slots).

Property 3: No connection request is starved. This property comes from the following theorem.

Theorem 3: *k*FRR serves an existing connection request within no more than $(\frac{N}{gk})^2$ cell slots.

Proof: The worse case service scenario of *k*FRR is the situation where a request from input port I_i to output port O_j has to wait all other $N/g - k$ input ports to be served by O_j, i.e. for some cell slot n, $Z_{i',j}(n) > 0$ (i.e. the number of cells in $Q_{i'j}$ at cell slot n) for all $I_{i'}$'s and $g_j = ((i+1) \bmod N/g)$, where $i' \ne i$. The delay between posting a request and serving the request consists of the delay for the request to be granted and the delay for the grant to be accepted. The delay for the request from I_i to O_j to be granted is

$(\frac{N}{gk}-1)\frac{N}{gk}$ cell slots since it takes $\frac{N}{gk}-1$ cell slots for O_j to grant requests from other $N/g-k$ input ports and it takes at most $\frac{N}{gk}$ cell slots for each grant to be accepted. After the grant to I_i is issued, it also takes $\frac{N}{gk}$ cell slots to get it accepted. Thus totally it takes $(\frac{N}{gk}-1)\frac{N}{gk}+\frac{N}{gk}=(\frac{N}{gk})^2$ cell slots to serve an existing connection request. ■

Property 4: kFRR finds a maximal size k-matching in at most $N/g-k+1$ iterations, i.e. kFRR converges in at most $N/g-k+1$ iterations.

The reason is explained as follows. The size of a maximal size k-matching is at most Nk/g. If finding a maximal size k-matching takes more than 1 iteration, the first iteration finds at least k^2 matches, the last iteration finds at least 1 match, and other iterations find at least k matches. Thus, the total number of iterations needed is at most $\lfloor\frac{Nk/g-k^2-1}{k}\rfloor+2$, which is given by $N/g-k+1$. We further conjecture that under uniform traffic arrivals kFRR converges in $O(\log N)$ iterations on average.

Fig. 6 shows an example of the number of iterations needed for kFRR to converge for an 8×8 SDMG CIOQ switch with $k=2$ under saturated load. In cell slot 0, kFRR takes 4 iterations to converge. It takes 3 and 2 iterations for kFRR to converge in cell slots 1 and 2 respectively. After cell slot 3, all grant pointers are totally desynchronized and kFRR converges in a single iteration.

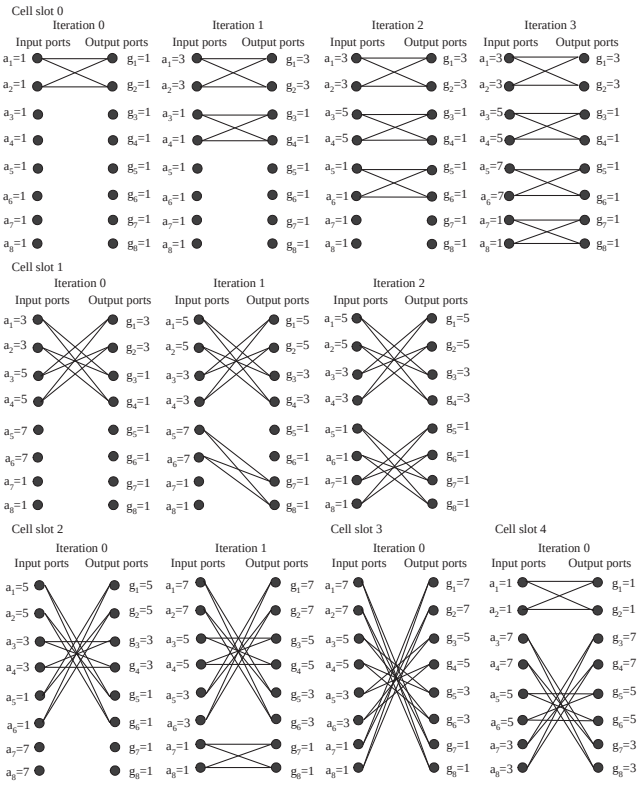

Fig. 6 Example of the number of iterations needed for kFRR to converge for an 8×8 SDMG CIOQ switch under saturated load.

6. Performance Evaluation*

In Section 4, we proved that any maximal size k-matching algorithms can achieve 100% throughput for SDMG CIOQ switches with an expansion factor 2. Nevertheless, in practice, the number of iterations allowed in one cell slot may not be sufficient for finding a maximal size k-matching. In this section, we evaluate the performance of kFRR with the number of iterations allowed in each cell slot is limited on SDMG CIOQ switches with an expansion factor 2 in terms of the average queuing delay. The queuing delay is defined as the cell's queuing delay at input and output ports counted in the number of cell slots.

Two traffic models are used in our simulations: uniform traffic and polarized traffic. For uniform traffic, we consider both Bernoulli arrivals and bursty arrivals. The polarized traffic is defined as follows [3]. Given the geometric progression factor $q\geq 1.00$, the proportion of traffic arriving at input line L_l destined for output line M_m should satisfy

$$d_{l,m}=\frac{q^{(l+m) \bmod N}\cdot(q-1)}{q^N-1}$$

such that,

$$\begin{array}{ll}\forall l\in[1..N], & \sum_{m=1}^{N} d_{l,m}=1 \text{ and}\\ \forall m\in[1..N], & \sum_{l=1}^{N} d_{l,m}=1.\end{array}$$

Polarized traffic with $q=1.00$ is uniform traffic. One can verify that both uniform traffic and polarized traffic satisfy the SLLN condition without oversubscribed input/output lines. Simulations have been performed for the kFRR algorithm for SDMG CIOQ switch sizes of 16×16, 32×32, 64×64, and 128×128 with different group factors (g), different port connection factors (k), different polarization factors (q), and different number of iterations. In the following, we present simulation results with the example of a 32×32 SDMG CIOQ switch. Without loss of generality, in our simulations, all pointers in kFRR are initialized randomly.

6.1 Bernoulli Arrivals

Fig. 7 shows the average cell delay vs. load of kFRR with 1, 2, and 4 iterations, $g=1$, $k=2$, and $q=1.00$, 1.50, and 2.00 for a 32×32 SDMG CIOQ switch under Bernoulli arrivals. In the figure, "x-y" represents the case of kFRR with $q=x$ and the number of iterations being equal to y. kFRR achieves 100% throughput for all polarization factors. The performance of kFRR improves when the polarization factor increases. We observe that the difference in the number of iterations does not affect much of the performance of kFRR under Bernoulli arrivals.

Fig. 8 compares the average queuing delay vs. load of one-iteration kFRR with $k=g$ (solid curve) and $k=2g$ (dotted curve) for $g=1,2$, and 4 for a 32×32 SDMG CIOQ switch under uniform Bernoulli arrivals. In the figure, "x-y" represents the case of kFRR with $g=x$ and $k=y$. kFRR with $k=2g$ improves the performance of kFRR with $k=g$ dramatically. For $k=2g$, larger group factor yields better performance.

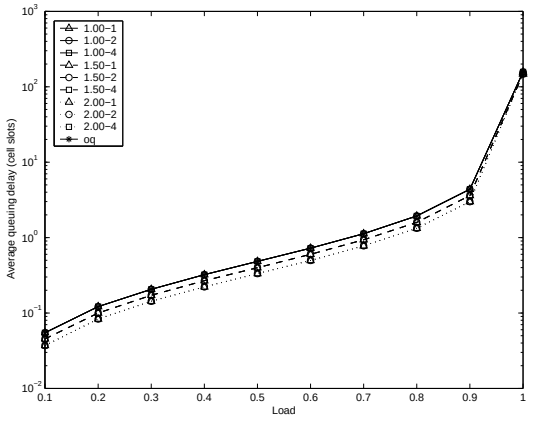

Fig. 7 Delay performance of kFRR with $g = 1$, $k = 2$, and different number of iterations under Bernoulli arrivals.

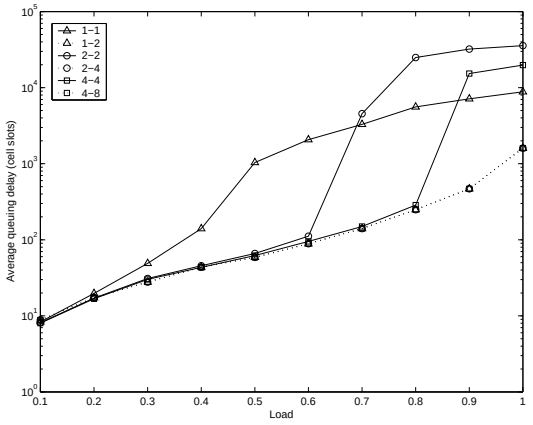

Fig. 10 Delay performance of one-iteration kFRR with different g's and different k's under bursty arrivals.

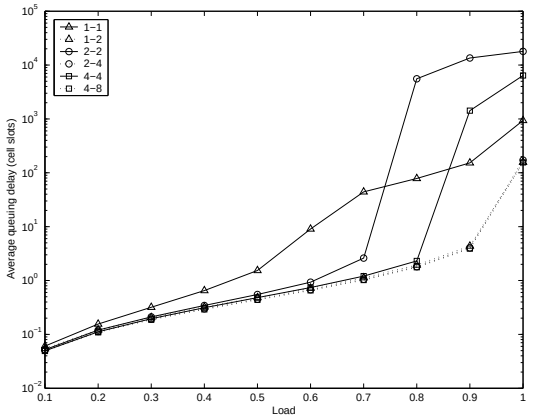

Fig. 8 Delay performance of one-iteration kFRR with different g's and different k's under uniform Bernoulli arrivals.

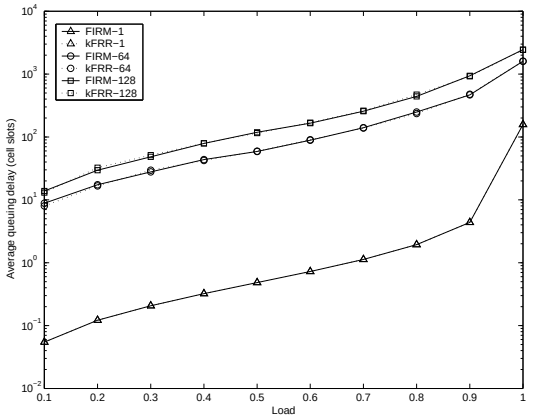

Fig. 11 Delay performance one-iteration kFRR with $P = 2$ and one-iteration FIRM with $S = 2$ for $g = 1$ under Bernoulli and bursty arrivals.

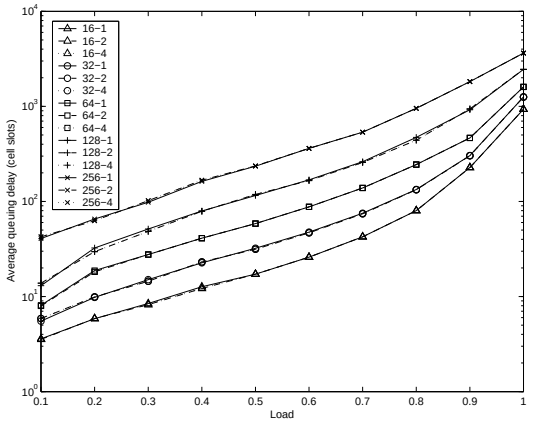

Fig. 9 Delay performance of kFRR with $g = 1$, $k = 2$, and different number of iterations under bursty arrivals.

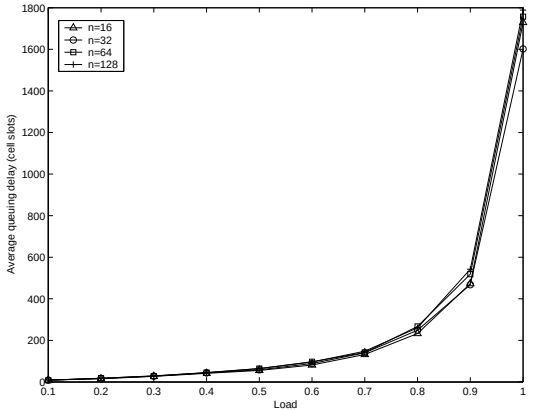

Fig. 12 Delay performance of one-iteration kFRR with different switch sizes under bursty arrivals.

6.2 Bursty Arrivals*

To show the performance of the proposed scheme under real traffic, such as multimedia traffic which tends to be bursty, we study the performance of kFRR under bursty traffic using 2-state markov-chain modulated on-off arrival processes [15], [16]. Each input line alternately generates a burst of full cells (all with the same destination) followed by an idle period of empty cells. The number of cells in each burst or idle period is geometrically distributed. Let $E(B)$ and $E(I)$ be the average burst length and the average idle length in the number of cells respectively. $E(I) = E(B)(1-\rho)/\rho$, where ρ is the load of each input line. We assume the destination of each burst is uniformly distributed. As a matter of fact, Bernoulli traffic can be considered as a special case of bursty traffic with $E(B) = 1$.

Fig. 9 illustrates the average queuing delay vs. load of kFRR with 1, 2, and 4 iterations, $g = 1$, and $k = 2$ for a 32×32 SDMG CIOQ switch under bursty arrivals with $E(B) = 16$, 32, 64, 128, and 256 respectively. In the figure, "x-y" represents the case of kFRR with $E(B) = x$ and the number of iterations being equal to y. kFRR achieves 100% throughput with all average burst length settings. The difference in the number of iterations does not affect much of the average queuing delay of kFRR under bursty arrivals.

Fig. 10 compares the average queuing delay vs. load of one iteration kFRR with $k = g$ (solid curve) and $k = 2g$ (dotted curve) for $g = 1, 2$, and 4 for a 32×32 SDMG CIOQ switch under bursty arrivals with $E(B) = 64$. In the figure, "x-y" represents the case of kFRR with $g = x$ and $k = y$. As shown in Fig. 10, kFRR with $k = 2g$ improves the performance of kFRR with $k = g$ dramatically. The performance of kFRR improves when the group factor increases.

Figure 11 compares the performance of one-iteration kFRR with $P = 2$ (solid curve) and one-iteration FIRM with $S = 2$ (dotted curve) for $g = 1$ of a 32×32 SDMG CIOQ switch under Bernoulli arrivals, bursty arrivals with $E(B) = 64$ and $E(B) = 128$. As we can see, under all cases, the performance of kFRR with expansion factor 2 achieves the same performance as FIRM with speedup factor 2.

6.3 With Different Switch Sizes

To evaluate the scalability of the SDMG CIOQ switch architecture and the kFRR algorithm, we have conducted the simulations for different switch sizes. Figure 12 shows the performance of kFRR with $g = 1$ and $k = 2$ for SDMG CIOQ switch sizes of 16×16, 32×32, 64×64, and 128×128. As shown in the figure, the delay performance of kFRR increases slightly with larger switch sizes. This confirms that the SDMG CIOQ switch architecture and the kFRR algorithm scale well as the switch size increases.

7. Hardware Implementation of kFRR Algorithm*

An important property of an efficient scheduling algorithm is simple to implement. In this section, we show that kFRR is ready to be implemented in hardware. Fig. 13 shows a possible design of a kFRR scheduler, which consists of $2N/g$ port arbitration components, a state update logic, and a state memory. Each port arbitration component (PAC) is responsible for selecting k out of Nk/g requests in a round-robin manner.

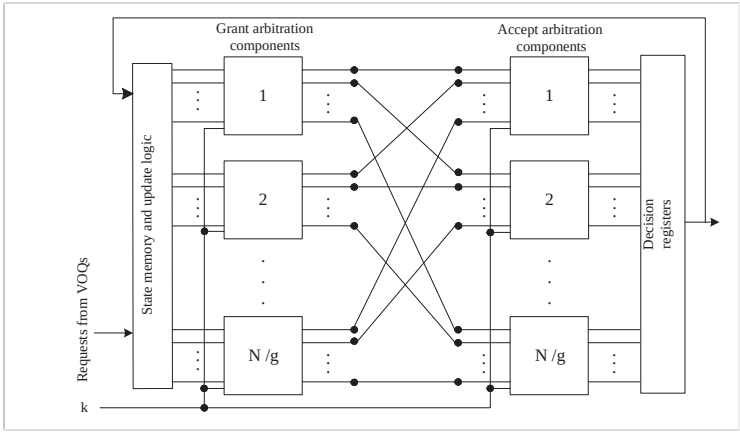

Fig. 13 Block diagram of a kFRR scheduler for an $N \times N$ SDMG CIOQ switch.

We discuss three possible designs of a PAC. The first design is employing the programmable priority encoder (PPE) proposed in [10]. The second design is using the parallel round-robin arbiter (PRRA) proposed in [29]. Since either the PPE or the PRRA can only make one selection each time, we have to run the PPE or PRRA k times to make k selections. The time needed for one-iteration kFRR using these two designs is $2k$ times the delay of an N/g-input PPE or PRRA. The third design is using the programmable k-selector proposed in [30]. The advantage of using programmable k-selectors is that the timing performance is independent of k. The time needed for one-iteration kFRR using such a design is 2 times the delay of an Nk/g-input programmable k-selector.

For an $N \times N$ SDMG CIOQ switch, the scheduler receives an $N/g \times \log k$-bit request vector from each input port at the start of each cell slot. Then, taking the example of one-iteration kFRR scheduler, it works as follows:

Step 1: Each grant PAC selects up to k requests and sends them to N/g accept PACs.

Step 2: Each accept PAC selects up to k grants and sends them to the decision register, the state memory and update logic, where the grant pointers are updated.

For an iterative kFRR scheduler, the PACs used are almost identical to those used for a one-iteration kFRR scheduler except the following differences. (1) The request matrix should be updated after each iteration. (2) The number of available connections at each PAC should be updated after each iteration. (3) Once an input/output port has no available connection, its PAC should be disabled in subsequent iterations of the same cell slot. These three modifications make an iterative kFRR scheduler slightly more complex than a one-iteration kFRR scheduler.

8. Concluding Remarks*

The major contributions of this chapter include: (1) We

introduced the SDMG CIOQ switch, which features space-division multiplexing expansion and grouped input/output ports to eliminate the speedup requirement of the switching fabric and memories of CIOQ switches. (2) We modelled the cell scheduling problem for the SDMG CIOQ switch as a bipartite k-matching problem. (3) Using fluid model techniques, we proved that any maximal size k-matching algorithm for the SDMG CIOQ switch with an expansion factor 2 can achieve 100% throughput so long as input line arrivals satisfy SLLN and no input/output line is oversubscribed. (4) We proposed an efficient and starvation-free distributed scheduling algorithm for the SDMG CIOQ switch, kFRR, for finding maximal size k-matchings. (5) Through simulations, we showed that kFRR achieves 100% throughput for the SDMG CIOQ switch with an expansion factor 2 for two SLLN traffic arrivals: uniform traffic and polarized traffic. (6) We proposed three hardware implementation schemes for the kFRR algorithm. In conclusion, the SDMG CIOQ switch provides an alternative solution to the CIOQ switch with speedup and kFRR is an efficient and practical scheduling algorithm for the SDMG CIOQ switch. Future work includes study of efficient scheduling algorithms supporting QoS differentiation for different types of traffic on the SDMG CIOQ switch.

References

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

[2] A. Awan and R. Venkatesan, "Design and implementation of enhanced crossbar CIOQ switch architecture," Proc. Canadian Conf. Eletrical and Computer Engineering, 2005, vol. 2, pp. 1045-1048.

[3] J. Blanton, H. Badt, G. Damm, and P. Golla, "Impact of polarized traffic on scheduling algorithms for high speed optical switches," presented on ITCom 2001, Denver, Aug. 2001.

[4] H. J. Chao, C. H. Lam, and X. L. Guo, "A fast arbitration scheme for terabit packet switches," Proc. GLOBECOM 1999, pp. 1236-1243.

[5] H. J. Chao, "Saturn: a terabit packet switch using dual round-robin," IEEE Commun. Mag., pp. 78-84, Dec. 2000.

[6] S. T. Chuang, A. Goel, N. Mckeown, and B. Prabhakar, "Matching output queuing with a combined input output queued switch," IEEE J. Select. Areas Commun., vol. 17, no. 6, pp. 1030-1039, Jun. 1999.

[7] W. J. Cook, W. R. Pulleyblank, A. Schrijver, and W. H. Cunningham, Combinatorial Optimization, John Wiley & Sons Inc., Nov. 1997.

[8] J. Dai and B. Prabhakar, "The throughput of data switches with and without speedup," Proc. INFOCOM 2000, pp. 556-564.

[9] D. Gale and L. S. Shapley, "College admission and the stability of marriage," American Mathematical Monthly, vol. 69, pp. 9-15, 1962.

[10] P. Gupta and N. Mckeown, "Designing and implementing a fast crossbar scheduler," IEEE Micro., vol. 19, no. 1, pp. 20-28, Jan.-Feb. 1999.

[11] J. E. Hopcroft and R. M. Karp, "An $n^{2.5}$ algorithm for maximum matching in bipartite graphs," Soc. Ind. Appl. Math. J., vol. 2, pp. 225-231, 1973.

[12] Y. Jiang and M. Hamdi, "A fully desynchronized round-robin matching scheduler for a VOQ packet switch architecture," Proc. IEEE HPSR 2001, pp. 407-412.

[13] M. J. Karol, M. G. Hluchyj, and S. P. Morgan, "Input vs. output queuing on a space-division packet switch," IEEE Trans. Commun., vol. 35, no. 12, pp. 110-115, May 1987.

[14] C. Li, S. Q. Zheng, and Mei Yang, "Scalable schedulers for high-performance switches," Proc. IEEE HPSR, 2004, pp. 198-202.

[15] S. Q. Li, "A general solution technique for discrete queuing analysis of multimedia traffic on ATM," IEEE Trans. Commun., vol. 39, no. 7, Jul. 1991.

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

[17] N. McKeown, A. Mekkittikul, V. Anantharam, and J. Walrand., "Achieveing 100% throughput in an input-queued switch," IEEE Trans. Commun., vol. 47, no. 8, Aug. 1999.

[18] C. Minkenberg, "On packet switch design," Ph.D. dissertation, Eindhoven University of Technology, 2001.

[19] H. Obara, S. Okamoto, and Y. Hamazumi, "Input and output queuing ATM switch architecture with spatial and temporal slot reservation control," Electronics Letters, vol. 28, no. 1, Jan. 1992.

[20] A. Pattavina, "Multichannel bandwidth allocation in broadband packet switch," IEEE J. Select. Areas Commun., vol. 6, no. 9, pp. 1489-1499, Dec. 1988.

[21] B. Prabhakar and N. McKeown, "On the speedup required for combined input and output queued switching," Automata, vol. 35, 1999.

[22] D. N. Serpanos and P. I. Antoniadis, "FIRM: a class of distributed scheduling algorithms for high-speed ATM switches with multiple input queues," Proc. INFOCOM 2000, pp. 548-555.

[23] D. Shah, "Maximal matching scheduling is good enough," Proc. GLOBECOM, 2003, pp. 3009-3013.

[24] I. Stoica and H. Zhang, "Exact emulation of an output queuing switch by a combined input output queuing switch," Proc. 6th IEEE/IFIP IWQoS 1998, pp. 218-224.

[25] R. E. Tarjan, Data Structures and Network Algorithms, Bell labs, Murray Hill NJ, 1983.

[26] Vitesse corp., TeraStream chip set [Online], Available: http://www.vitesse.com, 2003.

[27] M. Yang and S. Q. Zheng, "Pipelined maximal size matching scheduling algorithms for CIOQ switches," Proc. ISCC, 2003, pp. 521-526.

[28] M. Yang and S. Q. Zheng, "An efficient scheduling algorithm for CIOQ switches with space-division multiplexing expansion," Proc. IEEE INFOCOM, 2003, pp. 1643-1650.

[29] S. Q. Zheng, M. Yang, J. Blanton, P. Golla, and D. Verchere, "A simple and fast parallel round-robin arbiter for high-speed switch control and scheduling," Proc. 45th IEEE MWSCAS, 2002, pp. 671-674.

[30] S. Q. Zheng, M. Yang, and F. Masetti-Placci, "Constructing schedulers for high-speed, high-capacity switches/routers," Int. J. Computers and Applications, vol. 25, no. 4, pp. 264-271, 2003.

Appendix A: Proof of Lemma 2

Proof: Proving Equation (14) is equivalent to showing that, if $Z_{i,j}(n) \geq k$, then

$$C_{i,j}(n+1) - C_{i,j}(n)$$

$$\leq \sum_{j=1}^{N/g}(A_{i,j}(n+1)-A_{i,j}(n)) + \sum_{i=1}^{N/g}(A_{i,j}(n+1)-A_{i,j}(n)) - k. \quad (\text{A}\cdot 1)$$

Let $V_{i,j}$ denote the set of all VOQs holding cells arriving at input port I_i or destined for output port O_j. Then $C_{i,j}(n+1)-C_{i,j}(n)$ is the difference between the number of arrivals to $V_{i,j}$ at cell slot $n+1$ and the number of departures from $V_{i,j}$ at cell slot n. The number of arrivals to $V_{i,j}$ at cell slot $n+1$ equals to $\sum_{j=1}^{N/g}(A_{i,j}(n+1)-A_{i,j}(n)) + \sum_{i=1}^{N/g}(A_{i,j}(n+1)-A_{i,j}(n))$.

Since $Z_{i,j}(n) \geq k$ and the switch employs a maximal size k-matching algorithm, it follows from Equation (10) that

$$\sum_{j=1}^{N/g}\pi(n)_{i,j} + \sum_{i=1}^{N/g}\pi(n)_{i,j} \geq k.$$

That is to say that at least k cells are removed from those VOQ's in $V_{i,j}$. Thus, we get the bound on the right side of Equation (A$\cdot$1). ■

Appendix B: Proof of Theorem 2

Proof: To prove the theorem, we first show that the SDMG CIOQ switch is VOQ rate stable. In light of Theorem 1, this is equivalent to showing that the corresponding fluid model is VOQ weakly stable, i.e. every fluid solution (D,T,Z) has $Z(t)=0$ for $t \geq 0$.

Let E be the $N/g \times N/g$ matrix with each entry being 1. We have

$$C(t) = EZ(t) + Z(t)E, t \geq 0 \quad (\text{A}\cdot 2)$$

Define $f(t) = \langle Z(t), C(t)\rangle$, where $\langle A,B\rangle = \sum_{i,j} A_{i,j}B_{i,j}$. Then we have $f(t) \geq 0$ for $t \geq 0$ and $f(0)=0$. It is also true that $f(t)=0$ implies that $Z(t)=0$. We observe that

$$\begin{aligned} f(t) &= \sum_{i,j} Z_{i,j}(t)C_{i,j}(t) \\ &= \sum_{i,j} Z_{i,j}(t)(\sum_k Z_{i,k}(t) + \sum_k Z_{k,j}(t)) \\ &= \sum_{i,j,k}(Z_{i,j}(t)Z_{i,k}(t) + Z_{i,j}(t)Z_{k,j}(t)). \end{aligned}$$

Therefore,

$$\begin{aligned} \dot{f}(t) &= \sum_{i,j,k}\dot{Z}_{i,j}(t)Z_{i,k}(t) + \sum_{i,j,k} Z_{i,j}(t)\dot{Z}_{i,k}(t) + \\ &\quad \sum_{i,j,k}\dot{Z}_{i,j}(t)Z_{k,j}(t) + \sum_{i,j,k} Z_{i,j}(t)\dot{Z}_{k,j}(t) \\ &= 2\sum_{i,j,k} Z_{i,j}(t)\dot{Z}_{i,k}(t) + 2\sum_{i,j,k} Z_{i,j}(t)\dot{Z}_{k,j}(t) \\ &= 2\sum_{i,j} Z_{i,j}(t)\dot{C}_{i,j}(t) \\ &\leq 0, \end{aligned} \quad (\text{A}\cdot 3)$$

since from Lemma 2 and Equation (5),

$$\dot{C}_{i,j}(t) \leq \sum_{j=1}^{N/g}\lambda_{i,j} + \sum_{i=1}^{N/g}\lambda_{i,j} - k \leq g+g-2g = 0.$$

According to Lemma 1, $f(t)=0$ because $f(t)\geq 0$ and $\dot{f}(t) \leq 0$. Hence $Z(t)=0$ for $t \geq 0$, i.e. the fluid model is VOQ weakly stable. By Theorem 1, the SDMG CIOQ is VOQ rate stable, i.e., $\lim_{n\to\infty}\frac{D_{i,j}(n)}{n} = \lambda_{i,j}$. Also from Equation (5), the SDMG CIOQ switch is port conserving.

We then show that the SDMG CIOQ is work conserving. In fact, $\lim_{n\to\infty}\frac{D_{i,j}(n)}{n}$ is equal to the scheduled cell arrival rate from input port I_i to output port O_j, for any $1 \leq i,j \leq N/g$. Then we have the total scheduled cell arrival rate at output port O_j equals to

$$\sum_{i=1}^{N/g}\lim_{n\to\infty}\frac{D_{i,j}(n)}{n} = \sum_{i=1}^{N/g}\lambda_{i,j} = \sum_{l=1}^{N}\sum_{m=(j-1)g+1}^{jg}\lambda'_{l,m}.$$

Therefore, the scheduled cell arrival rate at output line M_m, where $(j-1)g+1 \leq m \leq jg$ for any $1 \leq j \leq N/g$, is given by

$$\lim_{n\to\infty}\frac{\sum_l D'_{l,m}(n)}{n} = \sum_{l=1}^{N}\lambda'_{l,m} \leq 1$$

based on Equation (2). Hence the SDMG CIOQ is work conserving, i.e. it can achieve 100% throughput if input line arrivals are sufficient. ■