

# $\frac{3}{2}$ —Approximations to Maximum Weight Matching Scheduling Algorithms for Networks of Input-Queued Switches

Claus Bauer, Dolby Laboratories, San Francisco, USA, cb@dolby.com

**Abstract**—It is well known that maximum weight matching algorithms guarantee the stability of input-queued switches, but are impractical due to their high computational complexity. In this paper, we investigate the application of matching algorithms that approximate maximum weight matching algorithms to scheduling problems. We show that while having a low computational complexity, they guarantee the stability of input queued switches when they are deployed with a moderate speedup.

Recent research has shown that scheduling algorithms that stabilize individual switches do not necessarily guarantee the stability of networks of input-queued switches. We apply recent results on networks of input-queued switches and show that the approximation algorithms proposed in this paper stabilize both single switches and networks of input-queued switches.

Finally, we show that the *improve\_matching* algorithm stabilizes networks of switches when it is deployed with a speedup of  $\frac{3}{2} + \epsilon$ .

## I. INTRODUCTION

Most high-speed switches are based on a combined input and output (CIOQ) queued architecture. As shown in figure 1, at each input the arriving cells are buffered into  $N$  Virtual Output Queues (VOQs) where each VOQ contains cells destined to a specific output. The switching core itself is modeled as a crossbar requiring that not more than one packet can be sent simultaneously from the same input or to the same output. The switching core works at a speed-up  $S \geq 1$ , which is defined as the ratio between the potentially higher speed of the switching core and the line-speed of the incoming (and outgoing) links.

The scheduling algorithm of a switch determines the stability and delay characteristics of the switch. McKeown [12] has shown that maximum weight matching (*MWM*) based scheduling algorithms that are deployed with a speedup of  $S = 1$  stabilize single input-queued switches when the weights are chosen proportional to the VOQ lengths. Due to the high complexity of  $O(N^3)$  of *MWM* algorithms, researchers have investigated less complex scheduling algorithms with performance characteristics similar to those of the *MWM* algorithm. It has been shown in [3], [6] that maximal weight matching algorithms stabilize a CIOQ switch when they are deployed with a speed-up of 2.

In this paper, we investigate the application of matching algorithms of low complexity that approximate *MWM* algorithms as scheduling algorithms for CIOQ switches. Generalizing the notion of a performance ratio from [9], we say that a matching algorithm approximates a *MWM* algorithm with approximation parameters  $(c, d)$ ,  $0 < c \leq 1$ ,  $d \geq 0$ , if for any values of the weights of the matching algorithm, the sum of

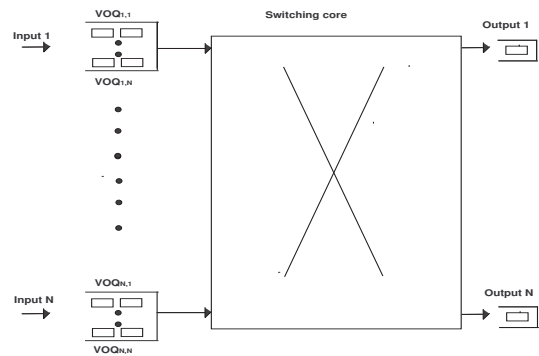


Fig. 1. Fig.1 Architecture of an input queued switch

the constant  $d$  and of the weight calculated by the matching algorithm is at least  $c$  times as large as the optimal weight calculated by the *MWM*.

First, we prove general results for matching algorithms that approximate a *MWM* algorithm with approximation parameters  $(c, d)$ . We discuss two modes to deploy these algorithms in a CIOQ switch. We show that in both modes, a deployment of the switch with a rational speed-up  $S \geq \frac{1}{c}$  is sufficient to guarantee the stability of a CIOQ switch.

In the next step, we apply our results to four known maximal matching algorithms that have a computational complexity of  $O(N^2)$ . We show that their deployment with a rational speed-up  $S \geq 2$  guarantees the stability of a CIOQ switch.

Finally, we discuss the new *improve\_matching* algorithm from [9] that approximates a *MWM* algorithm with approximation parameters  $(\frac{2}{3} - \epsilon, 0)$  and has a computational complexity of  $O(N^2)$ . Applying our general results on approximation algorithms, we show that this algorithm stabilizes a CIOQ switch when it is deployed a rational speedup of  $S \geq \frac{3}{2} + \epsilon$ .

The initial research on scheduling algorithms for CIOQ switches has only investigated the impact of scheduling algorithms on the stability of single switches. Researchers [4] have shown that scheduling algorithms that stabilize single switches might lead to instabilities when deployed in networks of input-queued switches. Stable switching policies for networks of switches that require the exchange of control information between switches in the network were introduced in [1], [4]. In [2] distributed scheduling policies based on the computationally complex *MWM* algorithms that require no communication between switches are proposed. Here, we

apply the ideas developed in [2] to the less complex approximation maximum weight matching algorithms and show that these algorithms stabilize single switches as well as networks of switches with moderate speed-up requirements. To the best knowledge of the author, this is the first time that stability for a network of input-queued switches could be shown for a non-MWM algorithms with a speedup strictly below 2.

In the next section, we introduce a mathematical model of a network of switches. In sections III and IV, we propose general scheduling algorithms based on approximation algorithms and prove stability results for these architectures. We describe stability results for specific approximation algorithms in section V and conclude in section VI.

## II. TERMINOLOGY AND MODEL

### A. Model of a network of queues

In this section, we follow an approach in [2] to describe a model of a queueing system. We assume a system of  $J$  physical queues  $\tilde{q}^j$ ,  $1 \leq j \leq J$  of infinite capacity. Each physical queue consists of one or more logical queues, where each logical queue corresponds to a certain class of customers within the physical queue. Whenever a packet moves from one physical queue to another, it changes class and therefore also changes logical queue. We denote a logical queue by  $q^k$ ,  $1 \leq k \leq K$ , where  $K \geq J$ . A packet enters the network via an edge switch, travels through a number of switches and leaves the network via another edge switch. We define a function  $L(k) = j$  that defines the physical queue  $\tilde{q}^j$  at which packets belonging to the logical queue  $q^k$  are buffered. The inverse function  $L^{-1}(j)$  returns the logical queues  $q^k$  that belong to the physical queue  $\tilde{q}^j$ .

Throughout this paper, the time  $t$  is described via a discrete, slotted time model. Packets are supposed to be of fixed size and an *external* time slot is the amount of time needed by a packet to arrive completely at an input link. For a speedup  $S \geq 1$ , we define an *internal* time slot as the amount of time needed for a packet to traverse the switching core.

We define a row vector  $X_n = (x_n^1, \dots, x_n^K)$ , where the  $k$ -th vector  $x_n^k$  represents the number of packets buffered in the logical queue  $q^k$  at the beginning of the  $n$ -th external time slot. We define  $E_n = (e_n^1, \dots, e_n^K)$ , where  $e_n^k$  equals the number of arrivals at the logical queue  $q^k$  in the  $n$ -th external time slot. Analogously, we define  $D_n = (d_n^1, \dots, d_n^K)$ ,  $0 \leq d_n^k \leq S$ , where  $d_n^k$  expresses the number of departed packets from  $q^k$  in the  $n$ -th external time slot. Thus, we can describe the dynamics of the system as follows:

$$X_{n+1} = X_n + E_n - D_n. \quad (1)$$

Packets that arrive at a logical queue  $q^k$  either arrive from outside the system or are forwarded from a queue within the system. Thus, we can write:

$$E_n = A_n + T_n, \quad (2)$$

where  $A_n = (a_n^1, \dots, a_n^K)$  denotes the arrivals from outside the system and  $T_n = (t_n^1, \dots, t_n^K)$  denotes the arrivals from inside the system.

We further define a routing matrix  $R = [r_{i,j}]$ ,  $1 \leq i, j \leq K$ ,

where  $r_{i,j}$  is the fraction of customers that depart from the logical queue  $q^i$  and are destined for the logical queue  $q^j$ . Assuming a deterministic routing policy, there holds,  $r_{i,j} \in \{0, 1\}$ ,  $\sum_{1 \leq i \leq K} r_{i,j} \leq 1$ ,  $\sum_{1 \leq j \leq K} r_{i,j} \leq 1$ . We set  $r_{i,j} \neq 0$ , if  $q^j$  follows  $q^i$  along the route. Noting that  $T_n = D_n R$  and writing  $I$  for the identity diagonal matrix, we find from (1) and (2):

$$X_{n+1} = X_n + A_n - D_n(I - R). \quad (3)$$

We assume that the external arrival processes are stationary and satisfy the Strong Law of Large Numbers. Thus,

$$\lim_{n \rightarrow \infty} \frac{\sum_{i=1}^n A_i}{n} = \Lambda \quad \text{w.p.1}, \quad (4)$$

where  $E[A_n] = \Lambda = (\lambda^1, \dots, \lambda^K)$ ,  $\forall n \geq 1$ .

We now calculate the average workload of the logical queues  $q^k$  which we denote by  $W = (w^1, \dots, w^K)$ . The expected traffic arriving from outside the system is by definition equal to  $\Lambda$ . The traffic that arrives at the logical queues after having passed through  $m$  previous queues inside the network is by the definition of the routing matrix  $R$  equal to  $\Lambda R^m$ . Noting that  $(I - R)^{-1} = I + R + R^2 + \dots$ , we find that the overall average workload at the logical queues  $q^k$  is given by  $W = \Lambda(I - R)^{-1}$ .

Finally, we give a stability criteria for a network of queues as proposed in [2].

**Definition 1:** A system of queues is rate stable if

$$\lim_{n \rightarrow \infty} \frac{X_n}{n} = \lim_{n \rightarrow \infty} \frac{1}{n} \sum_{i=0}^{n-1} (E_i - D_i) = 0 \quad \text{w.p.1.}$$

A necessary condition for the rate stability of a system of queues is that the average number of packets that arrive at any physical queue  $\tilde{q}^j$  during a timeslot is less than 1. In order to formalize this criteria, we introduce the following norm for a vector  $Z \in \mathbb{R}^K$ :

**Definition 2:** For a vector  $Z \in \mathbb{R}^K$ ,  $Z = (z^1, \dots, z^K)$ , and the function  $L^{-1}(k)$  as defined in this subsection, we set:

$$\|Z\|_{\max L} = \max_{j=1, \dots, J} \left\{ \sum_{k \in L^{-1}(j)} z^k \right\}. \quad (5)$$

If we apply this norm to the average workload vector  $W$ , then the expression  $\|W\|_{\max L}$  denotes the maximum average workload over all physical queues  $\tilde{q}^j$ . The necessary condition for rate stability can now be formalized as follows:

$$\|W\|_{\max L} < 1. \quad (6)$$

In the sequel, we will say that a system of networks that satisfies condition (6) and is rate stable achieves 100% throughput.

### B. Model of a network of switches

In this section, we apply the terminology of the previous section to a network of CIOQ switches. A network of CIOQ switches can be conceived as a queueing system as defined in the previous section where the virtual output queues are

<sup>1</sup>Throughout the paper, we abbreviate "with probability 1" by "w.p.1".

considered as the physical queues. In this model we neglect the output queues of the switches because instability can only occur at the *VOQs* (see [2]).

We say that packets that enter the network via the input of a given switch and leave the network via the output of a given switch belong to the same flow. Packets belonging to the same flow travel through the same sequence of physical queues and are mapped to the same logical queues at each physical queue, i.e., a flow can be mapped biunivocally to a series of logical queues.

We assume that each logical queue behaves as a FIFO queue and assume a *per-flow* scheduling scheme, which is more complex than a *per-virtual output queue* scheduling scheme. In section III we prove the main results of this paper for *per-flow* scheduling schemes. In [2], it has been shown how *per flow* scheduling schemes can be used to design *per-virtual output queue* scheduling schemes.

The network consists of  $B$  switches and each switch has  $N_b$ ,  $1 \leq b \leq B$ , inputs and outputs. If the total number of flows in the system is  $T$ , we do not have more than  $N_b^2$  physical queues and  $TN_b^2$  logical queues at switch  $b$ . We can model the whole network of switches as a system of  $\sum_{1 \leq b \leq B} TN_b^2$  logical queues. For the sake of simplicity, we suppose that  $N_b = N$ ,  $\forall b$ ,  $1 \leq b \leq B$  and set  $K = TN^2B$ . Finally, we define  $Q_I(b, i)$  as the set of indexes corresponding to the logical queues at the  $i$ -th input of the  $b$ -switch. Analogously,  $Q_O(b, i)$  denotes the set of indexes corresponding to the logical queues directed to the  $i$ -th output of the  $b$ -switch. We further note that logical queues are defined per switch, per virtual-output queue, and per flow. Thus, the index  $k$  of any logical queue in the network can be uniquely expressed as  $k = TN^2b + TNi + Tj + l$ ,  $0 \leq b < B$ ,  $0 \leq i, j < N$ ,  $0 \leq l < T$ . We use these definitions to adapt the norm  $\|Z\|_{maxL}$  to a network of switches that handle multiple flows at their inputs.

**Definition 3:** Given a vector  $Z \in \mathbb{R}^K$ ,  $Z = z^k$ ,  $k = TN^2b + TNi + Tj + l$ ,  $0 \leq b < B$ ,  $0 \leq i, j < N$ ,  $0 \leq l < T$  the norm  $\|Z\|_{IO}$  is defined as follows:

$$\|Z\|_{IO} = \max_{\substack{b=1, \dots, B \\ i=1, \dots, N}} \left\{ \sum_{m \in Q_I(b, i)} |z^m|, \sum_{m \in Q_O(b, i)} |z^m| \right\}.$$

Because we assume a deterministic routing policy, the necessary condition for rate stability given in (6) can now be written for a network of switches as follows:

**Definition 4:** For a network of CIOQ switches, a traffic and routing pattern  $W$  is admissible if and only if:

$$\|W\|_{IO} = \|\Lambda(I - R)^{-1}\| < 1. \quad (7)$$

Without further mentioning, in the rest of this paper, we will only consider traffic and routing patterns that satisfy the condition (7).

### III. APPROXIMATIONS TO THE *MWM* - ALGORITHM

In this section, we introduce local scheduling policies that are based on approximation algorithms. We present these policies in a non-distributed, centralized way, i.e., we will

assume that the configurations of all switches are computed by a centralized server, which sends each switch its specific configuration. Following an argument in [5], one can show how the algorithms presented in this paper can also be implemented in a distributed fashion, such that each switch calculates its own configuration. The stability results proved in this paper hold for a distributed implementation as well.

#### A. Weight function

All scheduling policies introduced in this paper are matching policies. Any matching policy is defined relative to a specific weight. For the definition of the weights, we will make use of a family of real positive functions  $f_k(x) : \mathbb{N} \rightarrow \mathbb{R}$ ,  $1 \leq k \leq K$ , that satisfy the following property:

$$\lim_{n \rightarrow \infty} \frac{f_k(n)}{n} = \frac{1}{w^k} \quad \text{w.p.1.} \quad (8)$$

We define  $\bar{d}^k(n) = \sum_{m \leq n} d_m^k$  as the cumulative number of services at queue  $q^k$  up to time  $n$ . Here, we assume that all switches start service at the same time  $m = 0$  and all switches continuously work until time  $n$ . We define the weight of the queue  $q^k$  at time  $n$  as

$$\phi_n^k = n - f_k(\bar{d}^k(n)). \quad (9)$$

This choice of the weights is motivated by the fact that it will allow us to derive the relation (25), which in turn is used for the proof of the rate stability in Theorem 1. We set  $\Phi_n = (\phi_n^1, \dots, \phi_n^K)$ . We note that the relations (8) and (9) imply that for any given positive integer  $v$ , there is

$$|\phi_n^k - \phi_{n+v}^k| \leq c_2(v), \quad (10)$$

where  $c_2(v)$  is a positive constant depending on  $v$ .

In [2], an example for  $f_k(n)$  is given. The cumulative function of external arrivals for the logical queue  $q^k$  is given by  $\bar{a}^k(n) = \sum_{m \leq n} a_m^k$ . The inverse function  $[\bar{a}^k]^{-1}(p)$  maps the packet number  $p$  to the arrival slot. Setting  $f_k(p) = [\bar{a}^k]^{-1}(p)$ , the weight  $\phi_n^k = n - [\bar{a}^k]^{-1}(p)$  denotes the age of the  $p$ -th packet at time  $n$ . At its departure time  $n$ , the age of the  $p$ -th packet is  $n - [\bar{a}^k]^{-1}(\bar{d}_n^k)$ .

#### B. Definition of an approximation algorithm

By definition [12], the choice of the matching  $D_n(MWM)$  by a *MWM* algorithm with weights  $P_n = (P_n^k)_{1 \leq k \leq B}$  in time slot  $n$  can be described by the relation

$$D_n(MWM)P_n^T = \max_{D_n} D_n P_n^T. \quad (11)$$

For a scheduling algorithm *ALGO* that approximates a *MWM* algorithm with approximation parameters  $(\frac{a}{b}, d)$ ,  $a, b \in \mathbb{N}$ ,  $a$  and  $b$  prime to each other, there holds by definition

$$D_n(ALGO)P_n^T \geq \frac{a}{b} D_n(MWM)P_n^T - d, \quad (12)$$

where  $D_n(ALGO)$  denotes the matching chosen by the algorithm *ALGO*. In this paper, we only consider approximation algorithms with weights chosen as in (9).

### C. The deployment of approximation algorithms in a switching core

We consider approximation algorithms that approximate the *MWM* algorithm with approximation parameters  $(\frac{a}{b}, d)$ . Without further mentioning it, we always assume that  $a \leq b$ ,  $a$  and  $b$  are prime to each other. To compensate for the factor  $\frac{a}{b}$ , we propose to deploy all approximation algorithms with a rational speed-up of  $S = \frac{b_1}{a_1} \geq \frac{b}{a}$ . We propose two different modes to implement an approximation algorithm in a CIOQ switch.

We extend the notation introduced in section II as follows: We define  $X_{n+\frac{da_1}{b_1}}$ ,  $0 \leq d \leq b_1 - 1$ , as the vector the entries of which are the number of packets buffered in the logical queues at time  $n + \frac{da_1}{b_1}$ . For every  $n$  satisfying  $n \equiv (0 \bmod a_1)$ , time  $n + \frac{da_1}{b_1}$  denotes the beginning of the  $(d+1)$ -th internal time slot after the  $n$ -th external time slot. Similarly,  $D_{n+\frac{da_1}{b_1}}$  expresses the number of packets departing in the  $(d+1)$ -th internal time slot of the  $n$ -th external time slot.

In the *mode\_keep*, the scheduling algorithm computes a matching at the beginning of a time slot  $n \equiv (0 \bmod a_1)$ . It keeps the matching constant until the beginning of the time slot  $n+a_1$ , when a new matching is calculated. In the interval  $[n, n+a_1)$ , up to cells  $b_1$  are forwarded at equally spaced time intervals of length  $\frac{a_1}{b_1}$ . For the *mode\_keep*, the evolution of the queue lengths is described as follows:

$$X_{n+\frac{da_1}{b_1}} = X_n - dD_n(\text{ALGO}) + \sum_{0 \leq c < \frac{da_1}{b_1}} E_{n+c} + D_\delta, \quad 1 \leq d \leq b_1,$$

where  $D_\delta = \max\left(\underline{0}, dD_n - X_n - \sum_{0 \leq c < \frac{da_1}{b_1}} E_{n+c}\right)$ . Here,

$\underline{0}$  is the vector with  $K$  elements with all entries equal to zero and and the maximum is taken for each matrix entry individually. If  $d_n^k = 1$ , the entry  $d_\delta^k$  denotes the difference between the number of cells that have been forwarded in the interval  $[n, n + \frac{da_1}{b_1})$ , and the number of internal time slots  $d$  in the interval. If  $d_n^k = 0$ , then  $d_\delta^k = 0$ .

In the *mode\_reconfig*, a new matching is computed in every internal time slot, i.e., every  $\frac{a_1}{b_1}$  external time slots, and cells are forwarded accorded to a calculated matching at most once. The queue evolution for the *mode\_reconfig* is described as follows:

$$X_{n+\frac{da_1}{b_1}} = X_n - \sum_{e=0}^{d-1} D_{n+\frac{ea_1}{b_1}}(\text{ALGO}) + \sum_{0 \leq c < \frac{da_1}{b_1}} E_{n+c} + G_\delta, \quad 1 \leq d \leq b_1,$$

where  $G_\delta$  is a vector with  $K$  elements where each entry is an integer between 0 and  $b_1$ . If at time  $n$ , there holds  $x_n^k \geq b_1$ , then there is  $g_\delta^k = 0$ . If at time  $n$ , there holds  $x_n^k < b_1$ , then depending on the *VOQ* length  $x_n^k$  at time  $n$  and the arrival patterns  $a_{n+c}^k$ ,  $0 \leq c \leq d-1$ , in the interval  $[n, n + \frac{da_1}{b_1})$ , the switch might not always be able to forward a packet from *VOQ* <sup>$i,j$</sup>  even if the scheduling algorithm prescribes so because

$d_{n+\frac{da_1}{b_1}}^k = 1$ . The value  $g_\delta^k$  equals the number of instances where this happens for the *VOQ* <sup>$i,j$</sup>  and thus takes values  $c$  in the range  $0 \leq c \leq b_1$ .

The *mode\_keep* mode requires less computations than the *mode\_reconfig* mode. However, the *mode\_reconfig* reacts faster to the changing lengths of the *VOQ*. Applying the analysis from [13], one can show that the *mode\_keep* mode leads to larger average package delays at the *VOQs* than the *mode\_reconfig* mode.

Now, we state the main result of this paper:

**Theorem 1:** For admissible traffic, a network of CIOQ switches that implements an approximation maximum weight matching algorithm with approximation parameters  $(\frac{a}{b}, d)$ , and which is deployed in either *mode\_keep* or *mode\_reconfig* with a rational speedup of  $S = \frac{b_1}{a_1} \geq \frac{b}{a}$  and where the weight  $\phi_n^k$  of queue  $q^k$  at time  $n$  is defined as in (9) achieves 100%.

## IV. PROOF OF THEOREM 1

### A. Lower bounds for the weights calculated by approximation algorithms

In this section, we define lower bounds for the weight calculated by an approximation algorithm deployed with a rational speedup  $\frac{b_1}{a_1} > \frac{b}{a}$  in either *mode\_keep* or *mode\_reconfig*. We will need these lower bounds for the proof of Theorem 1 in section IV-C. For our investigations, we consider the weight of all matchings calculated in  $a_1$  successive time slots  $[n, n+a_1)$  by an approximation algorithm with approximation parameters  $(\frac{a}{b}, d)$  in *mode\_keep*. Applying the relations (10) and (12), we find

$$\begin{aligned} & \sum_{d=0}^{b_1-1} D_n(\text{ALGO}) \Phi_{n+\frac{da_1}{b_1}}^T \\ & \geq b_1 D_n(\text{ALGO}) \Phi_n^T - b_1 K c_2(a_1) \\ & \geq \frac{ab_1}{b} D_n(\text{MWM}) \Phi_n^T - b_1 K c_2(a_1) - b_1 d \\ & \geq a_1 D_n(\text{MWM}) \Phi_n^T - b_1 K c_2(a_1) - b_1 d. \end{aligned} \quad (13)$$

Similarly, for an approximation algorithm with approximation parameters  $(\frac{a}{b}, d)$  that is deployed in *mode\_reconfig* with a rational speedup  $\frac{b_1}{a_1} > \frac{b}{a}$ , we find using the relations (10), (11), and (12):

$$\begin{aligned} & \sum_{d=0}^{b_1-1} D_{n+\frac{da_1}{b_1}}(\text{ALGO}) \Phi_{n+\frac{da_1}{b_1}}^T \\ & \geq \frac{a}{b} \sum_{d=0}^{b_1-1} D_{n+\frac{da_1}{b_1}}(\text{MWM}) \Phi_{n+\frac{da_1}{b_1}}^T - b_1 d \\ & \geq \frac{ab_1}{b} D_n(\text{MWM}) \Phi_{n+\frac{da_1}{b_1}}^T - b_1 d \\ & \geq a_1 D_n(\text{MWM}) \Phi_n^T - b_1 d - a_1 K c_2(a_1). \end{aligned} \quad (14)$$

### B. The fluid methodology

The proof of theorem 1 makes use of the fluid methodology as introduced in [6]. We recall an extension of the fluid model to a network of switches from [2] and refer the reader for further details to [2]. We define  $\Pi = \{\pi\}$  as the set

of all possible network-wide matchings, i.e. possible switch configurations throughout the network. Using (3), we obtain the fluid equations of the system as follows:

$$X(t) = X(0) + \Lambda t - D(t)(I - R), \quad (15)$$

$$D(t) = \sum_{\pi \in \Pi} \pi T_{\pi}(t), \quad (16)$$

$$\sum_{\pi \in \Pi} T_{\pi}(t) = t, \quad (17)$$

where  $T_{\pi}(t)$  is a non-decreasing function denoting the cumulative amount of time that the matching  $\pi$  has been used up to time  $t$ . Also, noting that by (8)  $\lim_{t \rightarrow \infty} f_k(t) \rightarrow t/w^k$ , and that  $\bar{d}^k(t) \rightarrow \infty$  for  $t \rightarrow \infty$ , we obtain

$$\phi^k(t) \rightarrow t - \frac{\bar{d}^k(t)}{w^k}. \quad (18)$$

### C. Proof of Theorem 1

We denote the set of all switch configurations found at time  $t$  by an approximation algorithm with approximation parameters  $(\frac{a}{b}, d)$  as defined in Theorem 1 as  $\Pi_{ALGO}(t)$ . The fluid equations (16) and (17) can be rewritten as:

$$\dot{D}(t) = \sum_{\pi \in \Pi_{ALGO}(t)} \pi_{ALGO} \dot{T}_{\pi_{ALGO}}(t), \quad (19)$$

$$\sum_{\pi_{ALGO} \in \Pi_{ALGO}(t)} T_{\pi_{ALGO}}(t) = t. \quad (20)$$

We define  $\Gamma = [\gamma^{(i,j)}]$  as the diagonal matrix with  $\gamma^{(k,k)} = w^k$ , and let  $\Gamma^{-1}$  be the inverse of  $\Gamma$ . Taking the derivative on both sides of (18), we obtain:

$$\dot{\Phi}(t) = \mathbb{I} - \dot{D}(t)\Gamma^{-1}. \quad (21)$$

Writing the scalar product for two vectors  $v_1$  and  $v_2$  as  $\langle v_1, v_2 \rangle = v_1 v_2^T$ , we define the Lyapunov function:

$$V(\Phi(t)) = \frac{1}{2} \langle \Phi(t)\Gamma, \Phi(t) \rangle.$$

We want to show that  $\forall t \geq 0$ ,

$$\Phi(t) \leq B, \quad (22)$$

for a certain constant  $B > 0$ . Noting that  $V(\Phi(0)) = \langle \Phi(0)\Gamma, \Phi(0) \rangle \geq 0$ , we see that if

$$\frac{d}{dt} V(\Phi(t)) \leq 0 \quad (23)$$

$\forall t$  such that  $\langle \mathbb{I}, \Phi(t) \rangle \geq K_0$  for a fixed  $K_0 > 0$ , then  $\forall t \geq 0$ , there holds  $V(\Phi(t)) \leq V(L_0) := \max_{\substack{L \in \mathbb{R}^K \\ \langle \mathbb{I}, L \rangle \leq K_0}} V(L)$ , which in

turn implies (22) for a certain  $B > 0$ . Thus we will show (23) in order to prove (22).

Finally, applying the principles of the fluid terminology as in [6] and dividing both sides of the equations by the number of considered time slots  $a_1$ , we can express the equations (13) and (14) in the fluid terminology in the following way:

$$\langle \pi_{MWM}(t), \Phi(t) \rangle \leq \langle \pi_{ALGO}(t), \Phi(t) \rangle + K_1, \quad \forall \pi_1 \in \Pi_1(t), \quad (24)$$

where  $\pi_{MWM}(t)$  and  $\pi_{ALGO}(t)$  are matchings chosen by the *MWM* and *ALGO* algorithms, respectively, and  $K_1 = (b_1 d + (a_1 + b_1) K c_2(a_1))/a_1$ . Now (23) follows from (19), (20), (21), and (24):

$$\begin{aligned} \frac{d}{dt} V(\Phi(t)) &= \frac{1}{2} \langle \dot{\Phi}(t)\Gamma, \Phi(t) \rangle + \frac{1}{2} \langle \Phi(t)\Gamma, \dot{\Phi}(t) \rangle \\ &= \langle \dot{\Phi}(t)\Gamma, \Phi(t) \rangle = \langle [\mathbb{I} - \dot{D}(t)\Gamma^{-1}]\Gamma, \Phi(t) \rangle \\ &= \langle W, \Phi(t) \rangle - \sum_{\pi_{ALGO} \in \Pi_{ALGO}(t)} \langle \pi_{ALGO} \dot{T}_{\pi_{ALGO}}(t), \Phi(t) \rangle \\ &\leq \langle W, \Phi(t) \rangle - \langle \pi_{MWM}, \Phi(t) \rangle + K_1 \\ &< 0, \end{aligned}$$

where the last estimate is derived using a well-known argument from [12]. We see from (18) and (22):

$$0 < t - \frac{\bar{d}^k(t)}{w^k} \leq B. \quad (25)$$

This implies  $\lim_{t \rightarrow \infty} \frac{\bar{d}^k(t)}{t} = w^k$ , i.e.,

$$\lim_{t \rightarrow \infty} \frac{D(t)}{t} = W, \quad \text{w.p.1}, \quad (26)$$

which corresponds to the rate stability condition of  $X(t)$  according to Definition 1.  $\square$

## V. EXAMPLES OF APPROXIMATION ALGORITHMS

### A. Maximal matching algorithms

The most common approximation algorithms to a *MWM* algorithm are variations of maximal matching algorithms. A maximal matching is a matching that is not properly contained in any other matching of the graph. Different types of greedy maximal matching algorithms are presented in [3], [14], and [7]. In [8], Drake and Hougardy propose a postprocessing step to further improve the performance of a given maximal matching. As all algorithms discussed in this section approximate any *MWM* algorithm with approximation parameters  $(\frac{1}{2}, d)$ , we deduce from Theorem 1:

*Theorem 2:* For admissible traffic, the approximation algorithms to a *MWM* algorithm presented in this section stabilize a network of CIOQ switches when they are deployed in either *mode\_keep* or *mode\_reconfig* with a rational speed-up  $S \geq 2$ .

Thus, among others, we provide a new way to prove that greedy maximal matching weight matching algorithms are stable with a rational speed-up of  $S \geq 2$  as shown in [3].

### B. The improve\_matching algorithm

We first give an overview of the main structure of the algorithm and refer the reader for the missing details to [9].

Using common notation, we denote a graph as  $G = (V, E)$ , where  $E$  and  $V$  are the set of edges and vertices of the graph, respectively. It is known from graph theory that for any given weight  $M$ , the weight of which is not the largest possible, one can replace some edges of  $M$  with some other edges of the graph such that the new set obtained is again a matching and has a strictly larger value than  $M$ . Such a process, which

```

Algorithm improve_matching
( $G = (V, E), w : E \rightarrow \mathbb{R}^+, M$ )

1   make  $M$  maximal
2    $M' := M$ 
3   for  $e \in M$  do begin
4     if there exists a  $\beta$ -augmentation in  $M'$  with
       with center  $e$ 
5     then augment  $M'$  by a good  $\beta$ -augmentation
       with center  $e$ 
6   end
7   return  $M'$ 

```

Fig. 2. Fig.2 The *improve\_matching* algorithm

adds a new set  $S \subset E$  to  $M$  and removes a set  $M(S)$  is called an augmentation. The set  $S$  is called the augmenting set for this augmentation and the set  $M(S)$  is defined as the set of all edges in  $M$  that are incident with an end vertex of an edge in  $S$ . If  $S$  contains edges of  $M$ , then these are also contained in  $M(S)$ . The gain of an augmentation is defined as  $w(S) - w(M(S))$  which is the increase of weight achieved by the augmentation.

The idea of the *improve\_matching* algorithm is first to use standard techniques to expand a given matching to a maximal matching (if the given matching is not already maximal) and then to make local improvements via appropriate augmentations to the maximal matching.

The *improve\_matching* algorithm considers only local improvements that are obtained via *short augmentations*. A *short augmentation* is defined as an augmentation such that all the edges in the augmenting set are adjacent to some edge  $e \in E$ . This edge  $e$  is called a center of this *short augmentation*. It may or may not belong to the actual matching which is locally changed. Examples of short augmentations are given in [9].

Furthermore, the algorithm does not consider all *short augmentations*, but only those *short augmentations* that lead to a local gain of a factor of at least  $\beta$ , where  $\beta$  is a fixed constant  $> 1$ . We call such a short augmentation that satisfies the inequality  $w(S) \geq \beta w(M(S))$  a  $\beta$ -augmentation and the augmenting set  $S$  the  $\beta$ -augmenting set. In a particular instant, there might be more than one possible  $\beta$ -augmentation. Intuitively, it is desirable to choose the  $\beta$ -augmentation that produces the biggest gain. However, for the purpose of the *improve\_matching* algorithm, it is sufficient to choose a *good  $\beta$ -augmentation*. A  $\beta$ -augmentation with center  $e$  is called *good* if it achieves at least  $(\beta - 1)/(\beta - \frac{1}{2})$  fraction of the gain that the best  $\beta$ -approximation with center  $e$  can achieve.

We now formally define the *improve\_matching* algorithm in figure 2. First, the input matching  $M$  is made maximal (if necessary) and then no further changes are made to  $M$ . Instead,  $M$  is copied to  $M'$  and all local augmentations are done with respect to  $M'$ . The algorithm visits each edge  $e \in M$  only once, and if it finds any  $\beta$ -augmentation set at this edge in  $M'$ , it performs a good  $\beta$ -augmentation centered at  $e$  in  $M'$ . In [9], it is shown that the complexity of the *improve\_matching* is linear in the number of edges  $E$ .

In order to achieve approximation parameters  $(\frac{2}{3} - \epsilon, 0)$  the

*improve\_matching* algorithm is applied iteratively. We first use a linear time maximal matching algorithm (see section V) to calculate a maximal matching  $M_0$  with weight at least as large as  $w(M_0) \geq \frac{1}{2}w(M_{opt})$ , where  $M_{opt}$  denotes a maximum weight matching of the graph  $G = (V, E)$ . We then apply the *improve\_matching* algorithm to the matching  $M_0$  to obtain a matching  $M_1$  and then iteratively apply the algorithm to the matching  $M_i$  to obtain a matching  $M_{i+1}$ . It is shown in [9] that at most  $O(\frac{1}{\epsilon})$  iterations are required to achieve approximation parameters  $(\frac{2}{3} - \epsilon, 0)$ . Thus, we deduce from Theorem 1:

*Theorem 3:* The iterated *improve\_matching* algorithm when it is deployed in either *mode\_keep* or *mode\_reconfig* with a rational speed-up  $S \geq \frac{3}{2} + \epsilon$  stabilizes a network of CIOQ switches under any admissible traffic.

## VI. CONCLUSIONS

This paper proposes the application of *MWM* - approximation algorithms scheduling algorithms for networks of input queued switches. We show that *MWM* approximation algorithms guarantee the stability of networks of switches under specific speedup requirements. Applying these results, we show that the *improved\_matching* algorithm guarantees the stability of a network of input-queued switches when it is deployed with a rational speed-up  $S \geq \frac{3}{2} + \epsilon$ .

## REFERENCES

- [1] Ajmone, M.M., Leonardi, E., Mellia, M., Neri, F., *On the throughput achievable by isolated and interconnected input-queued switches under multiclass traffic*, Proc. of Infocom 2002, New York City, June 2002.
- [2] Ajmone, M.M., Giaccone, P., Leonardi, E., Mellia, M., Neri, F., *Local scheduling policies in networks of packet switches with input queues*, Proc. of Infocom 2003, San Francisco, April 2003.
- [3] Ajmone M.M., Leonardi, E., Mellia, M., Neri, F., *On the stability of input-buffer cell switches with speed-up*, Proc. Infocom 2000, Tel Aviv, March 2000.
- [4] Andrews, M., Zhang, L., *Achieving stability in networks of input queued switches*, Proc. of Infocom 2001, Anchorage, Alaska, April 2001.
- [5] Bauer, C., *Distributed scheduling policies in networks of input-queued packet switches*, ACM Comp. Commun. Review, Vol. 34, no. 3, July 2004, 83 -92.
- [6] Dai, J.G., Prabhakar, B., *The throughput of data switches with and without speedup*, Proc. of IEEE Infocom 2000, Tel Aviv, March 2000.
- [7] Drake, D.E., Hougardy, S., *A simple approximation algorithm for the weighted matching problem*, Information Processing letters 85 (2003), 211-213.
- [8] Drake, D.E., Hougardy, S., *Linear time local improvements for weighted matchings in graphs*, WEA 2003, LNCS 2647, Seiten 107-119, 2003.
- [9] Drake, D. E., Hougardy, S., *A linear time approximation algorithm for weighted matchings in graphs*, preprint: <http://www.informatik.hu-berlin.de/~hougardy/paper.html>.
- [10] Gabow, H.N., Tarjan, R.E., *Faster scaling algorithms for general graph-matching problems*, J. ACM 38:4, 1991, 815-853.
- [11] Gabow, H.N., *Data structures for weighted matching and nearest common ancestors with linking*, SODA 1990, 434 - 443.
- [12] Keslassy, I., McKeown, N., *Achieving 100% throughput in an input queued switch*, IEEE Trans. on Communications, vol. 47, no. 8, Aug. 1999, 1260 - 1272.
- [13] Neely, M.M., Modiano, E., Rohrs, C.E., *Tradeoffs in Delay Guarantees and Computation Complexity for NxN Packet Switches*, Proc. of the Conf. on Information Sciences and Systems, Princeton: March 2002.
- [14] Preis, R., *Linear time 1/2 approximation algorithm for maximum weighted matching in general graphs*, Symposium on Theoretical Aspects of Computer Science (STACS) 1999, Springer LNCS 1563, 259 - 269.