Stable local scheduling policies in networks of input-queued switches

Claus Bauer, Research Scientist, Dolby Laboratories
100 Potrero Avenue, San Francisco, CA, 94103, USA
cb@dolby.com, tel.: +1 415 558 0343, fax: +1 415 863 1373
Keywords: Stable local scheduling policies, networks of switches

***Abstract*— Most research on switch architectures and scheduling algorithms has focused on single switches. It is well known that certain scheduling policies that are based on maximum weight matching algorithms guarantee the stability of single switches. However, recent research has shown that most of these scheduling policies do not guarantee the stability of networks of input-queued switches.**

So far, only one local scheduling policy has been proposed that provides the stability of a network of input-queued switches if it is deployed at every switch. In this paper, we show the existence of a large class of scheduling policies based on maximum weight matching policies that guarantee the stability of a network of switches. We also show the stability of networks of switches in which specific classes of different switching policies are deployed simultaneously at different switches.

It has been shown that the $iOCF$ algorithm guarantees the stability of a single input-queued switch if it is deployed with a speed-up of two. In this paper, we generalize the $iOCF$ algorithm to the $iNCOF$ algorithm and show under which condition the $iNOFC$ algorithm, when it is deployed without speedup, stabilizes networks of input-queued switches.

I. Introduction and Motivation

During the last decade, most research on switch architectures, scheduling algorithms and switch stability has focused on switch architectures and scheduling algorithms for single switches. Only very recently, researchers have started to investigate the stability of networks of switches.

With regard to single switches, input-queued switches have been investigated extensively. A typical input-queued switch is shown in figure 1. An $N \times N$ input-queued switch consists of N inputs and N outputs. It deploys buffers at the inputs and /or outputs. In order to avoid head of line blocking, each input buffer is split into N virtual output queues as follows: At each input i, packets that are destined for an output j, $1 \leq j \leq N$, are buffered in the virtual output queue $VOQ_{i,j}$. In general, the switching core is modeled as a crossbar such that not more than one packet can be sent simultaneously from the same input or to the same output. In addition, various switch architectures require that the switching speed in the switching core is S, $S \geq 1$ times faster than the speed of the input and output links. The factor S is denoted as the $speedup$.

At any moment in time, different packets might compete for input or output resources of the switch. Thus, the switch design includes the definition of a scheduling policy that decides which packets will be forwarded and which will be buffered. In particular, the designer is interested in a scheduling policy that guarantees the stability of the switch, i.e., that guarantees that the input and output queues do not grow infinitely under reasonable traffic arrival patterns.

For switches without speedup, a scheduling algorithm that guarantees the stability of the switch was first proposed in [11]. The scheduling problem is modeled as a bipartite maximum weight matching problem. The weights of the matching are the lengths of the $VOQs$, which we denote as $L_{i,j}$, $1 \leq i, j \leq N$. It was shown in [11] and later in [8], that if the average arrival traffic at each input and the average arrival traffic destined to each output is not more than 1, then this maximum weight matching policy guarantees the stability and even a bound on the average delay of the switch. In [12] and [14], the result in [11] has been generalized to a wider class of maximum weight matching algorithms. We define a real function $F(x) : \mathbb{R} \rightarrow \mathbb{R}$ which satisfies the following properties: I. F is monotonically non-decreasing and $F(0) = 0$, $F(x) >$ for $X > 0$, II. $\dot{F}(x)$ exists for all $X > 0$. It is shown in [14] that if we define a maximum weight matching policy that uses as weights the function values of the derivative of F, applied to the length of the $VOQs$, i.e. $\dot{F}(L_{i,j})$, then such a policy guarantees the stability of a single switch.

It was shown in [3] that in a specific network of input-

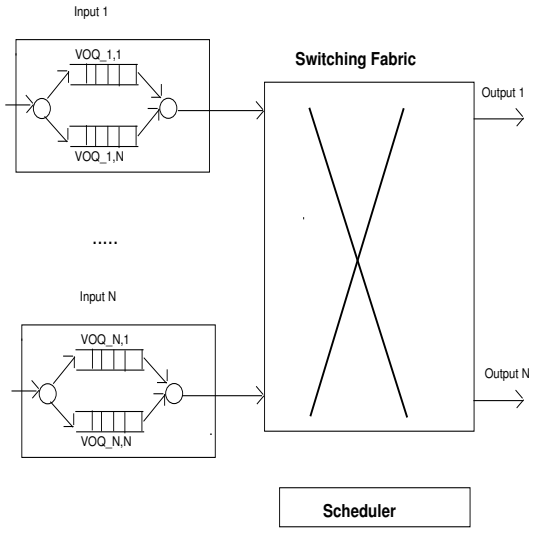

Fig. 1. Architecture of an input-queued switch

queued switches that is not overloaded, a maximum weight matching policy as defined in [11] that is implemented independently at each switch does not guarantee the stability of the network. The authors give a specific example where under non-overloaded conditions the number of packets buffered at specific VOQs at some switches in the network grows to infinity. This result showed that the previous work on maximum weight matching algorithms in isolated switches is of limited practical value in networks of switches.

In [1] and [3], switching policies are presented that guarantee the stability of all switches within a network, but require coordination and cooperation among the switches within the network. In [3], the LIN policy is proposed, which assumes that all switches in the network always know the actual traffic patterns all other switches. In [1], a stable scheduling policy that requires the exchange of state information only among adjacent switches is illustrated. Both policies require signaling traffic for the communication between the switches.

In [2], for the first time a *distributed* switching policy was proposed that guarantees 100% throughput in a network of input-queued switches. Each switch makes its scheduling decision independently of the other switches and no additional signaling traffic between the switches is required. The scheduling algorithm is a maximum weight matching algorithm with specific weights.

It is of interest to understand if other classes of distributed scheduling policies that guarantee the stability of a network of input-queued switches exist. In this paper, we will prove that the class of maximum weight algorithms defined in [14] and which was proven to be stable for an isolated switch can be modified to a local scheduling policy that guarantees the stability of a network of switches. We will also show that networks of switches that deploy different scheduling policies of this class at different switches simultaneously are stable.

Based on these results, we will also establish new theorems related to the theory of maximal matching algorithms and their deployment in networks of input-queued switches. Due to their low complexity, maximal matching algorithms have been widely researched ([5], [9], [10]). It has been shown in [6], [4] that maximal matching algorithms deployed with a speedup of two, or even slightly less, guarantee the stability of a switch. Recently, Shah ([14]) could prove that a specific maximal matching algorithm, the $iLQF$ algorithm which is defined as the maximal matching algorithm that uses the length of the $VOQs$ - the $L_{i,j}$ - as weights, is stable even without a speedup. This result assumes however, that ties are broken in a specific way or that no two VOQs have the same length.

Another maximal matching algorithm that has been widely researched in the literature ([5]) is the $iOCF$ algorithm, which is defined as the maximal matching algorithm that uses the time a cell has already spent in its respective VOQ as the weight.

Compared to the $iLQF$ algorithm, the $iOCF$ algorithm is more complex to implement as it requires the switch to attach a time tag to each cell. However, the $iOCF$ algorithm provides guarantees on the absolute packet delay which the $iLQF$ algorithm does not ([5]).

Thus, it is desirable to understand if the $iOCF$ algorithm can also provide stability without speedup. In this paper, we answer this question affirmatively. We not only show that the $iOCF$ algorithm is stable without speedup, but prove that the $iOCF$ algorithm can be generalized to the $iNOCF$ maximal matching algorithm that guarantees the stability of a network of switches when it is deployed without speedup.

The rest of the paper is organized as follows. Section II introduces the terminology to model networks of queues and networks of switches. In section III, we introduce the fluid methodology to describe the state of the network and define the classes of matching policies we are interested in. Then we prove that a network wide deployment of any of these matching policies guarantees the stability of a network. We also show that a network which deploys different matching policies simultaneously at different switches is stable. In section IV, we show that the $iNOCF$ algorithm ensures the stability of a network of input-queued switches when it is deployed without speedup. We conclude in section IV.

II. Terminology and Model

A. Model of a network of queues

In this section, we follow an approach in [2] to describe our 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 a timeslot is the time needed by a packet to arrive completely at an input link.

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

$$X_{n+1} = X_n + E_n - D_n. \tag{1}$$

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:

$$E_n = A_n + T_n,$$

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

$$X_{n+1} = X_n + A_n - D_n(I - R). \tag{2}$$

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

$$\lim_{n\to\infty} \frac{\sum_{i=1}^{n} A_i}{n} = \Lambda \qquad \text{w.p.1}, \tag{3}$$

where $E[A_n] = \Lambda = (\lambda^1, .., \lambda_K), \forall n \geq 1$[1]. Noting that $(I - R)^{-1} = I + R + R^2 + ...$, we find that the average workload $W = (w^1, ..., w^K)$ 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

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

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 definition:

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

$$||Z||_{maxL} = \max_{j=1,..,J} \left\{ \sum_{k\in L^{-1}(j)} z^k \right\}. \tag{4}$$

The necessary condition for rate stability can now be formalized as follows:

$$||W||_{maxL} < 1. \tag{5}$$

In the sequel, we will say that a system of networks that satisfies condition (5) 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 IQ/CIOQ switches. A network of IQ/CIOQ switches can be conceived as a queueing system as defined in the previous section where the virtual output queues are 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. It has been shown in [2] how stability results for *per-flow* scheduling schemes can be used to design stable *per-virtual output queue* 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 C, we do not have more than N_b^2 physical queues and CN_b^2 logical queues at switch b. We can model the whole network of switches as a system of $\sum_{1\leq b\leq B} CN_b^2$ logical queues. For the sake of simplicity, we suppose that $N_b = N, \forall b, 1 \leq b \leq B$ and set $K = CN^2B$. 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 use these definitions to adapt the norm $||Z||_{maxL}$ to a network of switches:

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

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

Because we assume a deterministic routing policy, the necessary condition for rate stability given in (5) can now be written for a network of switches as follows:

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

$$||W||_{IO} = ||\Lambda(I - R)^{-1}|| < 1. \tag{6}$$

In the rest of this paper, we will only consider traffic and routing patterns that satisfy the condition (6).

III. Maximum weight matching local scheduling policies

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 $f_k(x) : \mathbb{N} \to \mathbb{R}$, $1 \leq k \leq K$, that satisfy the following property:

$$\lim_{n\to\infty} \frac{f_k(n)}{n} = \frac{1}{w^k} \qquad \text{w.p.1.} \tag{7}$$

We define $\overline{d}^k(n) = \sum_{m\leq n} d_m^k$ as the cumulative number of services at queue q^k up to time n. For a given positive constant C, we define the weight of the queue q^k at time n as

$$\phi_n^k \quad = \quad n - f_k(\overline{d}^k(n)) + C. \tag{8}$$

[1]Throughout the paper, we abbreviate "with probability 1" by "w.p.1".

We set

$$\Phi_n = (\phi_n^1, .., \phi_n^K). \tag{9}$$

We see that for a fixed function $f(\cdot)$ that satisfies the relation (7), there exists a constant $C > 0$ such that $\forall n$, $n \geq 0$, there holds $f(\overline{d}_n^k) \leq \frac{\overline{d}_n^k}{w^k} - C$. Further, because the accumulative departure rate at each logical queue q^k cannot be more than the accumulative arrival rate, there holds $\lim_{n\to\infty} \frac{\overline{d}^k(n)}{w^k} \leq n$. Combining these two estimates, we see that for any function $f(\cdot)$ that satisfies the condition (7), one can always find a $C > 0$ such that the weights ϕ_n^k are positive $\forall n$, $n \geq 0$, $\forall k$, $1 \leq k \leq K$. This fact is important for the proof of theorem 1 below. In the sequel, we will always assume that the weights ϕ_n^k are positive.

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 $\overline{a}^k(n) = \sum_{m\leq n} a_m^k$. The inverse function $[\overline{a}^k]^{-1}(p)$ maps the packet number p to the arrival slot. Setting $C = 0$ and $f_k(p) = [\overline{a}^k]^{-1}(p)$, the weight

$$\phi_n^k = n - [\overline{a}^k]^{-1}(p) \tag{10}$$

denotes the age of the p-th packet at time n, i.e., it denotes the time the packet has already spent in the network. At its departure time n, the age of the p-th packet is $n - [\overline{a}^k]^{-1}(\overline{d}_n^k)$.

B. The fluid methodology

The main proof of this section will make use of the fluid methodology as introduced in [6], [7]. As in [2], we consider an extension of the fluid model to a network of switches. Based on the definitions introduced in section II A, we define three vectors as follows:
$X(t) = (X_1(t), ..., X_K(t))$ denotes the number of packets in the logical queues at time t, $D = (D_1(t), ..., D_K(t))$ denotes the number of packet departures from the logical queues until time t and $A = (A_1(t), ..., A_K(t))$ denotes the number of packets arrivals at the logical queues until time t. We define $\Pi = \{\pi\}$ as the set of all possible network-wide matchings and denote a specific scheduling algorithm by $\mathcal{F}$. For all $\pi \in \Pi$, we denote by $T_\pi^{\mathcal{F}}(t)$ the cumulative amount of time that the matching π has been used up to time t by the algorithm $\mathcal{F}$. Obviously, $T_\pi^{\mathcal{F}}(0) = 0$ $\forall \pi \in, \forall \mathcal{F}$. Using (2), we obtain the fluid equations of the system as follows:

$$X(t) = X(0) + \Lambda t - D(t)(I - R), \tag{11}$$

$$D(t) = \sum_{\pi\in\Pi} \pi T_\pi^{\mathcal{F}}(t), \tag{12}$$

$$\sum_{\pi\in\Pi} T_\pi^{\mathcal{F}}(t) = t. \tag{13}$$

The first equation models the evolution of the logical queues, whereas the second equation counts the total number of departures from the $VOQs$. The third equation reflects the fact that in each timeslot each input is connected to some output.

C. Maximum weight matching policies

In this section, we define a class of maximum weight matching policies that guarantees the stability of a network of input-queued switches. We introduce a set of functions $\mathcal{G}$ as follows:
Definition A real function F is said to belong to the set $\mathcal{G}$ if
a) F is monotonically non-decreasing, $F(0) = 0$, $F(x) > 0$ if $x > 0$.
b) $\dot{F}(x)$ exists for all $x > 0$.
Using the fluid methodology, we can define a function $\Phi(t)$ based on the definition of the function Φ_n in (9). We set

$$F_V(\Phi(t)) = \left(F(\phi^1(t), ..., F(\phi^K(t)\right).$$

We define $\Gamma = [\gamma^{(i,j)}]$ as the diagonal matrix with $\gamma^{(k,k)} = w^k$, and let Γ^{-1} be the inverse of Γ. Further, we write the scalar product for two vectors v_1 and v_2 as $\langle v_1, v_2\rangle = v_1 v_2^T$.

For every function $F \in \mathcal{G}$, we define a scheduling algorithm $MWM^{\mathcal{F}}$ as follows: At each time t, the scheduling algorithm $MWM^{\mathcal{F}}$ chooses the schedule $\pi^{\mathcal{F}}$ which is defined by the following equation:

$$\pi^{\mathcal{F}}(t) = \arg\max_{\pi} \left\{\langle \pi, \dot{F}_V(\phi(t))\rangle\right\}. \tag{14}$$

Finally, we define the Lyapunov function:

$$G(t) = \langle \mathbb{I}, \dot{F}_1(\Phi(t))\rangle,$$

where $F_1(x) = \Gamma F_V(x)$. Now, we can formulate the main result of this section:

Theorem 1: *For any real function $\mathcal{F} \in \mathcal{G}$, a network of IQ/CIOQ switches that implements a $MWM^{\mathcal{F}}$ achieves 100% throughput.*

D. Proof of theorem 1

We note that by (7) $\lim_{t\to\infty} f_k(t) \to t/w^k$, and that $\overline{d}^k(t) \to \infty$ for $t \to \infty$. Thus,

$$\phi^k(t) \to t - \frac{\overline{d}^k(t)}{w^k} + C. \tag{15}$$

We can write (15) as:

$$\dot{\Phi}(t) = \mathbb{I} - \dot{D}(t)\Gamma^{-1}. \tag{16}$$

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

$$\Phi(t) \leq B, \tag{17}$$

for a certain constant $B > 0$. We see that if

$$\frac{d}{dt} G(t) \leq 0 \tag{18}$$

$\forall t > 0$, then holds

$$G(t) \leq G(0),$$

which in turn implies (17) for a certain $B > 0$. Thus, we will show (18) in order to prove (17).
Now (18) follows from (12), (13), (14) and (16):

$$\begin{aligned}\frac{d}{dt}G(t) &= \frac{d}{dt}\langle 1, F_1(\Phi(t))\rangle \\ &= \sum_{k=1}^{K} w_k \dot{F}(\phi^k(t))\dot{\phi^k}(t) \\ &= \sum_{k=1}^{K} w_k \dot{F}(\phi^k(t))(1 - d_k w_k^{-1}) \\ &= \sum_{k=1}^{K} w_k \dot{F}(\phi^k(t)) - \sum_{k=1}^{K} d_k \dot{F}(\phi^k(t)) \\ &= \langle W, \dot{F}_V(\Phi(t))\rangle - \langle D(t), \dot{F}_V(\Phi(t))\rangle \\ &\leq 0.\end{aligned}$$

The relation (17) implies that:

$$0 < t - \frac{\overline{d}^k(t)}{w^k} + C \leq B. \tag{19}$$

Therefore,

$$\begin{aligned}\lim_{t\to\infty} \frac{\overline{d}^k(t)}{t} &= w^k, \qquad \text{i.e.,} \\ \lim_{t\to\infty} \frac{D(t)}{t} &= W, \qquad \text{w.p.1,}\end{aligned} \tag{20}$$

which corresponds to the rate stability condition for $X(t)$. □

IV. Networks of switches that deploy different scheduling policies

In the previous section, a class of scheduling policies for networks was presented. It was shown that each of these policies provides stability for a network of switches where all switches implement the respective scheduling policy.

It can now be asked if a network of switches in which each switch deploys any of those policies achieves 100% throughput. The answer is affirmative:

Theorem 2 *A network of IQ/CIOQ switches where each switch deploys a $MWM^{\mathcal{F}}$ policy for any $\mathcal{F} \in \mathcal{G}$ achieves 100% throughput.*

Proof: Assume that each switch in the network deploys any of the switching policies $MWM^{\mathcal{F}_1}$, .., $MWM^{\mathcal{F}_M}$. We divide the switches in the network into the M groups G_M, $i \in \{1, .., M\}$ where G_i contains the switches that deploy the switching policy $MWM^{\mathcal{F}_i}$. Accordingly, we can divide the departure vector $D(t)$ and the arrival rate vector W in M subvectors, i.e., we write $D(t) = (D_1(t), .., D_M(t))$ and $W = (W_1, .., W_M)$. In order to prove the stability condition (20), it is obviously sufficient to show that $\forall i \in \{1, .., M\}$

$$\lim_{t\to\infty} \frac{D_i(t)}{t} = W_i, \qquad \text{w.p.1.} \tag{21}$$

For each $i \in \{1, .., 4\}$, the relation (21) can be proved by applying the proof of theorem 1 to the corresponding group of switches G_i instead of applying them to the whole network of switches. □

V. Stability of the $iNOCF$ algorithm

The $iOCF$ maximal weight matching scheduling algorithm is described in [5] for a *per-virtual output queue* scheduling scheme. Here, we will expand this definition to a *per-flow* based scheduling scheme.

We recall that the switching core is modeled as a crossbar such that in every internal timeslot at most one packet can be sent from the same input or to the same output, i.e.,

$$||D_n||_{IO} \leq 1, \qquad \forall n \geq 1. \tag{22}$$

Further, we define $\forall b, i, j,\ 1 \leq b \leq B,\ 1 \leq i, j \leq N$,

$$\mathcal{S}_{b,i,j} := \left\{ m :\ 1 \leq m \leq K,\ m \in Q_I(b,i) \bigcup Q_O(b,j) \right\},$$

A maximal weight matching algorithm is defined for a set of positive weights P^k, $1 \leq k \leq K$, where P_k is the weight assigned to the logical queue q^k, as follows:

1) Initially, all logical queues q^k, $1 \leq k \leq K$, are considered potential choices for a cell transfer.
2) The logical queue with the largest weight, say q^{k_0}, is chosen for a cell transfer and ties are broken randomly. We assume without loss of generality that $k_0 \in Q_I(b_1, i_1)$ and $k_0 \in Q_O(b_1, j_1)$.
3) All logical queues q^k with $k \in \mathcal{S}_{b_1,i_1 j_1}$ are removed.
4) If all q^k are removed, the algorithm terminates. Else go to step 2.

Using this definition of a maximal weight matching algorithm, we now define the $iNOCF$ algorithm as the maximal matching algorithm that uses the weights defined in (10). These weights denote the time a packet has already spent in the network. Thus, we call the algorithm the $Network\,Oldest\,Cell\,First$ algorithm - $iNOCF$. If the network consists of only one switch, the $iNOCF$ algorithm is identical to the $iOCF$ algorithm as defined in [5].
We show the following theorem:

Theorem 3: Assuming that at any time t, no two weights $\dot{F}(\phi^k(t))$ are equal, the $iNOCF$ algorithm is stable.

Proof: We follow the proof of theorem 3 in [14]. If we choose $F(x) = e^x$, then we know from theorem 1 that the corresponding matching policy $MWM^{\mathcal{EXP}}$ guarantees the stability of a network of input queued switches. Thus, in order to show the stability of the $iNOCF$ algorithm under the assumption that at any point in time no two logical queues have the same weight, it is sufficient to show that under this assumption the $iNOCF$ and the $MWM^{\mathcal{EXP}}$ algorithm are identical. The identity of both algorithms is proved in exactly the same way as lemma 3 in [14]. We note that the proof in [14] requires that the weights are always integers. The definition of the weights in (10) satisfies this requirement. The assumption that no two logical queues have the same weight is realistic for large queue sizes due to the randomness of packet arrivals.□

VI. Conclusions

In this paper, we investigate local scheduling policies for networks of input-queued switches. We show that there ex-

ists a large class of switching policies that guarantee the stability of a network of input-queued switches. It is also proved that a network of input-queued switches where each switch implements any scheduling policy out of this class of policies achieves 100% throughput. Finally, it is shown that the $iNOCF$ maximal weight matching algorithm can stabilize a network of input-queued switches under the assumption that no two logical queues have the same weight at any time.

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] Andrews, M., Zhang, L., *Achieving stability in networks of input queued,* Proc. of Infocom 2001, Anchorage, Alaska, April 2001.

[4] Benson, K., *Throughput of crossbar switches using maximal matching algorithms,* Proc. of IEEE ICC 2002, New York City.

[5] Charny, A., Krishna, P., et al., *Algorithms for providing bandwidth and delay guarantees in input buffered crossbars with speedup,* Proc. of IWQoS, Napa, CA, 1998.

[6] Dai, J.G., Prabhakar, B., *The throughput of data switches with and without speedup,* Proc. of IEEE Infocom 2000, Tel Aviv.

[7] Dai, J.G., *Stability of fluid and stochastic processing networks,* Miscellanea publication n.9, Centre for Mathematical Physics and Stochastic, Denmark (http://www.maphysto.dk), Jan. 1999.

[8] Leonardi, E., Mellia, M., Neri, F., Marsan, M.A., *Bounds on average delay and queue size averages and variances in input-queued cell-based switches,* Proc. of IEEE Infocom 2001, Anchorage, Alaska.

[9] Leonardi, E., Mellia, M., Neri, F., Marsan, M.A., *On the stability of input-buffered cell switches with speed-up,* Proc. of IEEE Infocom 2001, Anchorage, Alaska.

[10] Leonardi, E., Mellia, M., Neri, F., Marsan, M.A., *Stability of maximal size matching scheduling in input queued cell switches,* Prof. of IEEE ICC 2000, New Orleans.

[11] McKeown, N., Mekkittikul, A., Anantharam, V., Walrand, J., *Achieving 100% throughput in an input queued switch,* Proc. 39th Annual Allerton Conference on Communication, Control and Computing, Oct. 2001.

[12] Keslassy, I., McKeown, N., *Achieving 100% throughput in an input queued switch,* IEEE Transactions on Communications, vol. 47, no. 8, Aug. 1999, 1260 - 1272.

[13] Shah, D., Kopikare, M., *Delay bounds for approximate maximum weight matching algorithms for input queued switches,* Proc. of IEEE Infocom 2002, New York City, June 2002.

[14] Shah, D., *Stable Algorithms for input-queued switches,* Proc. 39th Annual Allerton Conference on Communication, Control and Computing, Oct. 2001.