Local Scheduling Policies in Networks of Packet Switches with Input Queues

M. Ajmone Marsan, P. Giaccone, E. Leonardi, F. Neri

***Abstract*— A significant research effort has been devoted in recent years to the design of simple and efficient scheduling policies for Input Queued (IQ) and Combined Input Output Queued (CIOQ) packet switches. As a result, a number of switch control algorithms have been proposed. Among these, scheduling policies based on Maximum Weight Matching (MWM) were identified as optimal, in the sense that they were proved to achieve 100% throughput under any admissible arrival process satisfying the strong law of large number.**

On the contrary, it has been recently shown that the usual MWM policies fail to guarantee 100% throughput in *networks* of interconnected IQ/CIOQ switches. Hence, new policies suited for networks of interconnected switches were proposed and proved to achieve 100% throughput. All of these new policies require coordination and cooperation among different switches.

In this paper we address the open problem of the existence of *local* scheduling policies that guarantee 100% throughput in a network of IQ/CIOQ switches, providing a positive answer to such question. The only assumptions on the input traffic are that it satisfies the strong law of large numbers and that it does not oversubscribe any link in the network.

I. Introduction and Previous Work

Input Queueing (IQ) switch architectures have become an attractive architectural solution for the design of large-size and high-capacity packet switches. Anderson [1] and McKeown [2] showed in their seminal works that the negative effects of Head-of-the-Line (HoL) blocking on performance can be reduced or completely eliminated by adopting per-destination queueing (also called Virtual Output Queueing – VOQ) at input cards, and by controlling the switch operations with a scheduling algorithm. The scheduling algorithm avoids internal switch conflicts by deciding which input port is enabled to transmit packets without conflicts through the switching fabric.

We refer in this paper to fixed-size data units, called “cells” from the ATM jargon, possibly obtained by segmenting variable-size packets (for example IP datagrams), and to a synchronous switch operation, according to which input/output permutations are changed synchronously at every cell time (called “slot”) for all switch ports.

The problem faced by scheduling algorithms with virtual output queues can be formalized as a maximum size or maximum weight matching on the bipartite graph in which nodes represent input and output ports, and edges represent cells to be switched. Edges may be associated with weights related to the state of input queues. For an isolated switch dealing with a single traffic class, scheduling policies implementing at each slot a Maximum Weight Matching (MWM) with edge weights proportional to input queue lengths were proved to achieve the same performance in terms of throughput of an Output Queueing (OQ) switch in [3], [4], and [5], under a wide class of traffic patterns. The best known implementations of MWM exhibit a computational complexity $O(P^3)$, where P is the number of switch ports. To achieve good scalability in terms of switch size and port data rate, it is essential to reduce the computational complexity of the scheduling algorithm. This objective has been often pursued by introducing a moderate speed-up with respect to the data rate of input/output lines [6] in the switching fabric, as well as in the linecard memories. In this case, buffering is required at outputs as well as inputs, and the term “combined input/output queueing” (CIOQ) is used.

A wide set of results are known also for CIOQ switching architectures. CIOQ switches with speed-up equal to 2 have been proved to be able to exactly emulate OQ switches implementing any monotonic work-conserving queueing discipline [6]. This result holds under general traffic conditions also for switches interconnected to other switches; its practical relevance, however, is strongly limited by the very large complexity required to implement the scheduling policy. A wide class of low complexity scheduling policies, among which maximal size matching algorithms (see Sect. IV), have been proved in [4] and [7], with speed-up equal to 2, to achieve the same throughput performance of OQ switches. The best known implementations of maximal size matching exhibit a computational complexity $O(P^2)$.

In [8] it was shown that a specific network of IQ switches implementing a MWM scheduling policy can exhibit an unstable behavior when switches are not overloaded. This new, counterintuitive result, opened new perspectives in the research on IQ and CIOQ switches, reducing the value of most of the results obtained for switches in isolation. In [8] the authors propose a policy named LIN that, if implemented in each switch of the network, leads to 100% throughput under any admissible traffic pattern when each traffic flow in the network is leaky-bucket compliant. The LIN policy, however, is based on a pre-scheduling of cell transmissions at each switch of the network, thus relying on an exact knowledge of the traffic pattern at each switch (which asks for a large signaling bandwidth), and leading to excessive computational complexity when the traffic load approaches 1.

In [9] we elaborated on the findings of [8], proposing a stable scheduling policy that requires the exchange of state information only among adjacent switches.

This work was supported by the MIUR PLANET-IP Project. The authors are with Dipartimento di Elettronica, Politecnico di Torino, Italy; {surname}@mail.tlc.polito.it

In this paper we address the open (to the best of the authors' knowledge) problem of the existence of *local* scheduling policies that guarantee 100% throughput in a network of IQ/CIOQ switches, i.e. scheduling policies that do not require information about the states of other switches.

Two methodologies were applied in the past to obtain most of the known theoretical results on IQ and CIOQ switches: the Lyapunov function methodology, and the fluid models methodology. The Lyapunov function methodology was applied in [3], [7], [5], to find the stability region of several scheduling algorithms under general traffic patters. The fluid models methodology [10], [11], was used in [4] for the same purpose.

Results in this paper are obtained by applying both the Lyapunov function and the fluid models methodologies. The interested reader can refer to [9] for a presentation of the basic theoretical results that form the background necessary to our analysis.

After introducing our notation and assumptions in Sect. II, we introduce in Sect. III stable scheduling policies for networks of IQ and CIOQ switches that avoid any exchange of state information among switches. In Sect. IV, instead, we introduce semi-local stable scheduling policies, which require only a limited amount of signaling among switches. Sect. VI concludes our paper. To ease a first reading of the paper, we moved all proofs and heavier notations to appendices.

II. Notation and Definitions

A. Queueing Systems

Consider a system of J discrete-time (physical) queues $\tilde{q}^{(j)}$ of infinite capacity, each identified by an index j, with $0 \leq j < J$. The system of queues handles $N \geq J$ classes of customers. Each customer arrives at the network from the outside, receives service at a number of queues, and leaves the network. Customers change class every time they move through the network, since they cross different logical queues. We suppose that each class k of customers, $0 \leq k < N$, univocally identifies a queue in the system at which all class k customers are enqueued, i.e., all customers of class k are enqueued at the same queue. Let $L(k) = j$ be the system location function that associates each class k of customers with the queue j at which class k customers are enqueued. $L^{-1}(j)$ is the counter-image of j through function $L(k)$ and returns the set of customer classes served at queue j. When $N = J$, each customer class is in one-to-one correspondence with a queue.

Let $X_n = (x_n^{(0)}, x_n^{(1)}, \dots, x_n^{(N-1)})$ be the row vector whose k-th component $x_n^{(k)}$, $0 \leq k < N$, represents the number of customers of class k in the system at time n. We say that the set of customers of the same class forms a logical queue in the system of queues; thus in the paper we indicate the set of customers of class k with the term "logical queue k". Logical queue k is denoted by $q^{(k)}$. Each physical queue $\tilde{q}^{(j)}$, $0 \leq j < J$, therefore comprises the set $L^{-1}(j)$ of logical queues. We suppose that the service times required by customers of all classes are deterministic and equal to one unit of time.

The evolution of the number of queued customers is described by $x_{n+1}^{(k)} = x_n^{(k)} + e_n^{(k)} - d_n^{(k)}$, where $e_n^{(k)}$ represents the number of class k customers that entered logical queue k (and thus physical queue $L(k)$) in time interval $(n, n+1]$, and $d_n^{(k)}$ represents the number of customers departed from logical queue k in time interval $(n, n+1]$. $E_n = (e_n^{(0)}, e_n^{(1)}, \dots, e_n^{(N-1)})$ is the vector of entrances in the logical queues, and $D_n = (d_n^{(0)}, d_n^{(1)}, \dots, d_n^{(N-1)})$ is the vector of departures from the logical queues. With this notation, the system evolution equation can be written as:

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

The entrance vector is sum of two terms: vector $A_n = (a_n^{(0)}, a_n^{(1)}, \dots, a_n^{(N-1)})$ representing the customers arrived at the system from outside, and vector $T_n = (t_n^{(0)}, t_n^{(1)}, \dots, t_n^{(N-1)})$ of recirculating customers; $t_n^{(k)}$ is the number customers departed from some logical queue and entered into logical queue k in time interval $(n, n+1]$. Note that when customers do not traverse more than one queue (as it is typically the case for a switch in isolation), vector T_n is null for all n, and $A_n = E_n$.

The $N \times N$ matrix $R = [r_{ij}]$ is the *routing matrix,* whose element r_{ij} represents the fraction of customers departing from logical queue i and reaching logical queue j. We assume deterministic routing, hence R is a binary doubly sub-stochastic matrix (i.e., $r_{ij} = \{0, 1\}$, $\sum_i r_{ij} \leq 1$, $\sum_j r_{ij} \leq 1$); $r_{ij} \neq 0$ if queue $q^{(j)}$ follows $q^{(i)}$ along the customer route.

Note that $T_n = D_n R$. The law of evolution of logical queues can thus be rewritten as:

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

where I is an identity diagonal matrix.

Let us consider the external arrivals process $A_n = (a_n^{(0)}, a_n^{(1)}, \dots, a_n^{(N-1)})$; we suppose that arrival processes are stationary, i.e., $E[A_n] = \Lambda = (\lambda^{(0)}, \lambda^{(1)}, \dots, \lambda^{(N-1)})$ does not depend on the time interval $[n, n+1)$. Moreover, we suppose that arrival processes at each logical queue satisfy the Strong Law of Large Numbers (SLLN), i.e.:

$$\lim_{n \to \infty} \frac{\sum_{i=0}^{n-1} A_i}{n} = \Lambda \qquad \text{with probability } 1$$

The average workload W provided at each logical queue by customers that entered the system of queues in time interval $[n, n+1)$ is given on average by $W = \Lambda(I - R)^{-1}$, since $(I - R)^{-1} = I + R + R^2 + \dots$.

B. Norms and Other Operators

Before proceeding, we define two norm functions that will be helpful in the sequel.[1]

Definition 1: Given a vector $Z \in \mathbb{R}^N$, $Z = (z^{(k)}, 0 \leq k < N)$, we call $||Z||_2$ the Euclidean Norm of Z, i.e., $||Z||_2 = \sqrt{\sum_{k=0}^{N-1} \left(z^{(k)}\right)^2}$.

[1]In this paper, $\mathbb{N}$ denotes the set of non-negative integers, $\mathbb{R}$ denotes the set of real numbers, and $\mathbb{R}^+$ denotes the set of non-negative real numbers. Furthermore "with probability 1" will be abbreviated into "w.p.1"

***Definition** 2:* Given a vector $Z \in I\!R^{+N}$, $Z = (z^{(k)}, \ 0 \leq k < N)$, and a location function $L(k) = j$, from $0 \leq k < N$ to $0 \leq j < J$, with $J \leq N$, norm $||Z||_{\mathrm{maxL}}$ (the name refers to maximum queue length) is defined as:

$$||Z||_{\mathrm{maxL}} = \max_{j=0,\dots,J-1} \left\{ \sum_{k \in L^{-1}(j)} z^{(k)} \right\} \tag{3}$$

C. Stability Definitions for a System of Queues

Several definitions of stability for a network of queues can be found in the technical literature. We refer in this paper to *rate stability*.

***Definition** 3:* A system of queues is *rate stable* if

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

where X_n is the queue lengths vector at time n.

A necessary condition for the system of queues to achieve rate stability is that the average workload provided at each physical queue by customers entering the system of queues in time interval $[n, n+1)$ does not reach 1. This condition, that we call *no-overload condition*, is also a sufficient condition for rate stability in any BCMP type network of queues [12]. This condition can be formalized as:

$$||W||_{\mathrm{maxL}} < 1 \tag{4}$$

In general, as shown in [10], [11], [8], [9], this condition does not guarantee the stability of a generic network of queues.

A system that is rate stable under any workload meeting the no-overload condition is said to achieve *100% throughput*.

D. Notation for Networks of Switches

We introduce the notation for a network of S IQ/CIOQ cell-based switches, which extends some of the previous definitions. We refer to the case of fixed-size packets, often called cells. Switch s, $0 \leq s < S$, has P_s input ports and P_s output ports, all running at the same cell rate.

We say that packets that enter the network at a given input port of a given switch, and leave the network from a given output port of a given switch, belong to the same *(information) flow.* The packets belonging to the same information flow behave as customers in the queueing system of Section II-A: they traverse a sequence of physical queues from the network ingress point to the network egress point. At each physical queue, customers belonging to a given flow are mapped onto one logical queue. In other words, a sequence of different logical queues is biunivocally associated with a flow. Under deterministic routing, all logical queues belonging to the same flow receive the same workload. We normally assume that each *physical* queue is handled as a FIFO queue, while some state information may have to be maintained for each logical queue mapped onto that physical queue.

When instead each logical queue behaves as a FIFO, i.e., customers arrive and depart from logical queues, we have a different, more complex queue management, called *per-flow queueing.*

Each switch adopts a Virtual Output Queuing (VOQ) scheme at the inputs: one physical FIFO queue is maintained at each input for each output. As noted above, each physical queue is partitioned in a number of logical queues, associating one logical queue with each flow traversing the physical queue. If C is the number of information flows, at switch s we have P_s^2 physical queues (VOQs), and up to CP_s^2 different logical queues.

For network of CIOQ switches under admissible traffic, since it can been shown that instabilities may originate only at input queues, we neglect the output queueing process[2].

The network of IQ/CIOQ switches can be thus modeled as a system of $N = \sum_s CP_s^2$ logical queues (at the inputs), each of them being identified by a unique index. We restrict our study to the case $P_s = P \ \forall s$, so that $N = CP^2S$ and $J = P^2S$. Let $Q_I(s,i)$ be the set of indexes corresponding to the logical queues of switch s at input port i. Similarly, let $Q_O(s,j)$ be the set of indexes corresponding to the logical queues of switch s directed to output port j.

We can adapt the definition of $||Z||_{\mathrm{maxL}}$ to the case of a network of switches handling multiclass traffic, as follows:

***Definition** 4:* Given a vector $Z \in I\!R^{+N}$, $Z = \{z^{(k)}, k = CP^2s + CPi + Cj + l, \ 0 \leq s < S, \ 0 \leq i,j < P, \ 0 \leq l < C\}$, the norm $||Z||_{IO}$ is defined as:

$$||Z||_{IO} = \max_{\substack{k=0,\dots,S-1 \\ i=0,\dots,P-1}} \left\{ \sum_{k \in Q_I(k,i)} |z^{(k)}|, \sum_{k \in Q_O(k,i)} |z^{(k)}| \right\}$$

If vector Z refers to the workload in a network of switches, $||Z||_{IO}$ becomes the load on the most loaded input or output port.

At each time slot, a set of non contending cells departs from the VOQ of each switch. More formally, we say that:

***Property** 1:* At each time slot, the departure vector $D \in \{0,1\}^N$ satisfies:

$$||D||_{IO} \leq 1$$

***Definition** 5:* Let $W = [w^{(k)}] = \Lambda(I - R)^{-1}$ be the effective load vector. The traffic pattern loading a network of IQ/CIOQ switches is admissible if and only if:

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

Remember that, with deterministic routing, the workload is the same for all the logical queues along a given flow route in the network.

Thanks to Bramson's result [13] on the stability of a generic queueing network adopting FIFO service policies, it is possible to state the following:

***Property** 2:* A network of OQ switches adopting a FIFO service policy achieves 100% throughput.

[2]The formal proof relies on the fact the fluid limit of a rate stable queue is an empty queue as better explained in [10]

Note that, because of contentions inside switches, a network of IQ/CIOQ switches cannot adopt a pure FIFO service policy and Bramson's result cannot be applied.

III. Stable local policies in networks of IQ/CIOQ switches

In the following sections we consider two classes of policies, the first based on Birkhoff-von-Neumann decomposition and the second on the maximum weight matching algorithm.

A. *Birkhoff-von-Neumann based policies*

In Theorem 1 we discuss the properties of a generic local policy that achieves 100% throughput in a network of IQ/CIOQ switches. We show that a Birkhoff-von-Neumann (BvN) decomposition of the workload matrix locally at each switch is sufficient to achieve 100% throughput. The drawback of this approach is that it requires the knowledge of the arrival rates, and this motivates Theorem 2, which shows that 100% throughput can be achieved also by using the BvN decomposition together with an estimation of the arrival rates.

The following definitions introduce some classes of scheduling policies which we will show to achieve 100% throughput.

Definition 6: Consider a scheduling policy $\mathcal{P}$ and a vector $\Gamma = [\gamma^{(i)}]$ of size N such that $||\Gamma||_{IO} < 1$. $\mathcal{P}$ is (t_0, Γ)-compliant if, w.p.1, there exists a t_0 and a finite F such that in every temporal window $[t, t+F]$, with $t > t_0$, $\mathcal{P}$ guarantees an average service rate greater than $\gamma^{(i)}$ to every queue $q^{(i)}$ that is constantly backlogged [3].

Definition 7: Consider a scheduling policy $\mathcal{P}$ and a vector $\Gamma = [\gamma^{(i)}]$ of size N such that $||\Gamma||_{IO} < 1$. $\mathcal{P}$ is (t_0, Γ)-weak-compliant if, w.p.1, there exists a t_0 and a finite F such that in every temporal window $[t, t+F]$, with $t > t_0$, $\mathcal{P}$ guarantees an average service rate greater than $\sum_{k \in L^{-1}(j)} \gamma^{(k)}$ to every queue $\tilde{q}^{(j)}$ that is constantly backlogged.

Definition 8: A scheduling policy $\mathcal{P}$ is Γ-compliant if it is $(0, \Gamma)$-compliant; it is Γ-weak-compliant if it is $(0, \Gamma)$-weak-compliant.

Theorem 1: Let W be the workload on the network of switches. Any W-compliant (and (t_0, W)-compliant, with t_0 finite) scheduling policy guarantees 100% throughput in any network of IQ/CIOQ switches, whenever the ingress stochastic processes satisfy the SLLN. The same holds for any W-weak-compliant (and (t_0, W)-weak-compliant) scheduling policy.

We report the proof of this theorem in Appendix I.

The problem to be solved can be summarized in the following way: Do local W-compliant (or (t_0, W)-compliant) scheduling policies exist in networks of IQ/CIOQ switches? If yes, how can a W-compliant (or (t_0, W)-compliant) scheduling policy be constructed?

We can consider two cases: W is known and W is unknown.

Given the knowledge of any admissible workload W, a W-compliant scheduling policy can be always obtained according to a BvN decomposition [3], [14]. Indeed, the BvN result states that any vector W such that $||W||_{IO} \leq 1$ can be written as a convex combination of admissible departure vectors, i.e., for each W with $W = \sum_i \alpha_i D_i$ with $\alpha_i \geq 0$ and $\sum \alpha_i = 1$, where D_i are admissible departure vectors, i.e. $\|D_i\|_{IO} \leq 1$.

Thus, given an admissible vector W (i.e., a vector with $||W||_{IO} < 1$), it is always possible to obtain $W' = \frac{W}{||W||_{IO}}$, which can be decomposed in a convex combination of departure vectors according to the BvN theorem.

A policy that selects, at each time slot, a departure vector D_i according to a Weighted Round-Robin [15] (WRR) with weights α_i can be shown to be W-compliant. Rounding problems in the case of non rational α_i can be made negligible by choosing a window size F large enough. Note that this WRR-based policy can be locally implemented if every switch selects its own departure vector according to a BvN decomposition of its local traffic. We call this policy W-driven Birkhoff-von-Neumann (W-BvN) scheduling policy.

However, in this case, vector W must be a-priory known. Unfortunately, in most applications the workload W is unknown. Thus, the next question becomes: Is there a way to locally estimate in a dynamic (on-line) fashion the information on W? We provide a positive answer in the next theorem.

Theorem 2: Given any network of IQ/CIOQ switches, a BvN policy driven by an on-line estimation $\hat{W}_n = [\hat{w}_n^{(k)}]$ of W achieves 100% throughput under any admissible traffic pattern if $\hat{W}_n \to W$ w.p.1. This can be obtained if the estimation is performed according to the rule:

$$\hat{w}_n^{(k)} = \frac{1}{n} \sum_{i=0}^{n-1} e_i^{(k)} \tag{5}$$

We report the proof of this theorem in Appendix II.

By additivity of the limits, also the on-line estimation of the arrival rate at the physical queues converges to the workload of the physical queues. Hence, when the BvN policy is applied directly on the physical queues, it achieves 100% throughput.

B. *Maximum weight matching based policies*

The BvN approach still has two drawbacks: i) it requires the implementation of BvN decomposition at each switch, which is costly; ii) being the policy insensitive to the queues state, the delay performance can be poor. Moreover its reaction to traffic changes can be slow.

It may thus be appealing to devise more reactive local policies that guarantee 100% throughput. Bearing this goal in mind, we first investigate MWM scheduling policies. We wonder if there is a way to assign weights to queues based on local informations so that any network of IQ/CIOQ switches achieves 100% throughput under any admissible traffic pattern. Also in this case the answer is positive.

Theorem 3: A network of IQ/CIOQ switches implementing a MWM scheduling policy in which the weight $\phi_n^{(k)}$ of queue $q^{(k)}$ at time n is computed as:

$$\phi_n^{(k)} = n - f(d_n^{(k)})$$

[3]More formally, there exists an $\epsilon > 0$ such that the average service rate is greater than $\gamma^{(i)} + \epsilon$

achieves 100% throughput if:

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

We report the proof of this theorem in Appendix III. From this last result, it is possible to devise a stable local policy, which is equivalent to the LIN policy proposed in [8]:

***Corollary** 1:* A network of IQ/CIOQ switches implementing a N-OCF (Network Oldest Cell First) scheduling policy, where the weights are the age of packets inside the network, achieves 100% throughput.

Proof: Define $\bar{a}_n^{(k)}=\sum_{m\le n}a_m^{(k)}$ the cumulative function of the external arrivals for queue k. $\bar{a}_n^{(k)}$ can be viewed as a function of (discrete) time n, with image set corresponding to packet ordinal numbers. Call $[\bar{a}_n^{(k)}]^{-1}$ the inverse function, mapping packet numbers to their (arrival) time slots[4]. The age of the p-th packet inside the network observed at time n at queue k is simply $n-[\bar{a}_n^{(k)}]^{-1}(p)$. The age of the p-th packet at its departure time $d_n^{(k)}$ is $n-[\bar{a}_n^{(k)}]^{-1}(d_n^{(k)})$. Observe now that:

$$\lim_{p\to\infty}\frac{[\bar{a}_n^{(k)}]^{-1}(p)}{p}=\frac{1}{w^{(k)}} \qquad \text{w.p.1}$$

because the external arrival process satisfies the SLLN; now apply Theorem 3. ■

An interesting corollary of this theorem provides a result which does not refer to a network of switches, but is an extension of the one in [3], showing that MWM using age of cells at the switch as weights achieves 100% throughput under admissible i.i.d. Bernoulli traffic processes.

***Corollary** 2:* A single switch *in isolation* implementing OCF, i.e., a MWM with weights equal to the age of cells at queue heads, is rate stable when the arrival process is admissible and satisfies the SLLN.
The proof is immediate, since it is a rephrased version of Corollary 1, when the network comprises one single switch.

We now propose a scheduling algorithm, called *Approximate Oldest Cell First* (A-OCF), that is able to achieve 100% throughput because of Theorem 3.

A-OCF Algorithm. Recall that physical queue $\tilde{q}^{(j)}$, $0\le j<J$, comprises the set $L^{-1}(j)$ of logical queues. At time slot n, the algorithm works in the following way (note that the algorithm is local to each switch):

1) for each queue $\tilde{q}^{(j)}$, $0\le j<J$, update the estimate of the average queue load:

$$\tilde{w}_n^{(j)}=\sum_{k\in L^{-1}(j)}\hat{w}_n^{(k)}$$

according to new arrivals, using (5);
2) assign a tag $g_p^{(j)}$ to queue $\tilde{q}^{(j)}$; tags are initialized to 0. If the queue has experienced the departure of the p-th packet at time $n-1$, tags are updated according to the formula:

$$g_{p+1}^{(j)}=g_p^{(j)}+1/\tilde{w}_n^{(j)}$$

3) if queue $\tilde{q}^{(j)}$ is non-empty, associate a weight equal to $\max(0,n-g^{(j)})$; otherwise, its weight is equal to 0;
4) compute the MWM with weights as above, and configure the switching fabric.

[4]Care must be paid in inverting $\bar{a}_n^{(k)}$: we skip here mathematical details to improve readability

Let us now derive the computational complexity of A-OCF. Considering the whole network, Steps 1 and 2 require $O(J)$ operations, Step 3 requires $O(J)$ operations, whereas Step 4 requires $O(SP^3)$ operations, since each MWM costs $O(P^3)$. Considering the single switch, the previous complexity figures should be divided by S.

To conclude this section, we report two conjectures, that we formulate, but we were unable to prove.

***Conjecture** 1: A network of IQ/CIOQ switches in which each switch implements a MWM scheduling policy whose weights are cell ages achieves 100% throughput.*

If proved, this can be seen as an extension of Bramson's result [13], which proved that every generalized Kelly network [17] of queues, in which each queue implements a FIFO service policy, is rate stable.

***Conjecture** 2: A policy in which at each switch the departure vector is made on average proportional to the queue lengths vector achieves 100% throughput.*

IV. Stable semi-local policies in networks of CIOQ switches

We now present a class of semi-local policies for interconnected IQ/CIOQ switches that guarantee 100% throughput.

By semi-local policy we mean a scheduling policy that requires some form of coordination among the switches in the network, but a small amount of signaling, since it can be implemented in a distributed fashion at different switches on the basis of (dynamic) local information, and of (static) topological information.

For this purpose, we define the concept of *logical-queues dependency graph*:

***Definition** 9:* The logical-queues dependency graph is a direct graph $\mathcal{G}_D=G(V,E)$ whose vertexes correspond to logical queues. A direct edge $v_i\to v_j$ corresponding to queues q_i and q_j exists if the state of queue q_j depends in some way the state of q_i. In particular, an edge $v_i\to v_j$ exists if:

- q_j immediately follows q_i on the route of some flow. In this case packets leaving q_i enter q_j modifying its state, thus, the scheduling decision at q_i impacts the state of q_j.
- q_i and q_j are conflicting queues, located at the same switch, and thus the selection of a packet from q_i prevents the selection of a packet from q_j.

Note that, if the scheduling policy at the switch in which q_i and q_j are located is such that packets from q_j are selected only when q_i is empty (strict priority system), then no edge exists from v_j to v_i in the dependency graph, since the service at q_i is not affected by the state of q_j.

Although it is easy to observe that, in a generic network of IQ/CIOQ switches, the logical-queues dependency graph $\mathcal{G}_D$ can be cyclic, the following nice property holds when graph G_D is acyclic:

***Theorem** 4:* Given a network of IQ/CIOQ switches satisfying the following two properties:

1) every switch in the network implements a scheduling policy that locally guarantees 100% throughput (i.e., that would guarantee 100% throughput if the switch were isolated);
2) every switch implements a scheduling policy such that the logical-queues dependency graph is acyclic;

then the network of switches achieves 100% throughput.
We report the proof of this theorem in Appendix IV.

We must therefore face the problem of designing a scheduling policy that achieves 100% throughput at every switch in isolation, while keeping acyclic the logical-queues dependency graph. We partition the logical queues in C sets $C^{(i)}$, with $0 \leq i < C$, where C is the number of packet flows traversing the network. $C^{(i)}$ comprises all the logical queues storing packets of the i-th flow in the network. This is called *per-flow partition*.

The problem can be transformed in the definition of a scheduling policy such that: i) logical queues are served according to strict priorities depending on the per-flow partition: queues belonging to set $C^{(i)}$ are given strict priority over queues belonging to set $C^{(j)}$, with $i < j$; ii) it locally achieves 100% throughput.

The latter property cannot be obtained with speed-up 1, since stable local policies rely on the application of a MWM algorithm whose weights are related to queue lengths or cell ages, which is incompatible with the defined strict priority scheme.

However, it is possible to define maximal-size-matching[5] (called "mSM") policies that are compatible with the strict priority scheme defined above, thanks to the fact that CIOQ switches with mSM and speed-up 2 were proved to achieve 100% throughput. Thus any network of interconnected CIOQ switches with speed-up 2 can be made stable if the scheduling policy implemented at each switch is a mSM, that is compatible with the priority scheme based on the per-flow partition as defined above.

Since a speedup equal to 2 is adopted, two matchings are computed at each time slot. Within a single switch, the algorithm used to compute each matching is called *Prioritized maximal Size Matching* (P-mSM):

P-mSM Algorithm. The policy runs through the following steps:

1) set all the inputs and outputs as unmatched; set $k = 0$;
2) compute a mSM only among the logical queues belonging to $C^{(k)}$ and set as matched all the input/output pairs of the matching;
3) if any non-empty input (or output) remains unmatched, increment k by one and repeat from Step 2; otherwise, stop.

[5]A maxim*um* matching is one with the highest weight/size, whereas a maxim*al* matching is a matching to which it is impossible to add edges. A maximum matching is always maximal, but not viceversa.

At the end of P-mSM, a maximal matching is computed among logical queues; this dictates switch configurations. Note that the P-mSM algorithm can be split in local sub-problems: the maximality of the matching at a switch guarantees the maximality of the overall matching. The only non-local informations required is a unique flow identification; moreover, a unique agreement is required on the flow priority assignment.

Thanks to Theorem 4 we can state the following:

***Corollary** 3:* Consider a network of CIOQ switches with speedup 2. P-mSM with per-flow partition achieves 100% throughput.

In each switch, the complexity of this algorithm is $O(CP^2)$, since the complexity of an $P \times P$ mSM is $O(P^2)$. Unfortunately, complexity grows with the number of flows, which is unacceptable from a practical point of view.

This motivates us to extend the definition of cyclic dependency graph to devise more practical stable semi-local policies:

***Definition** 10:* A logical-queues dependency graph is *semi-acyclic* if, for any cycle $(e_{i_1}, e_{i_2}, \dots)$, all the corresponding queues $(q_{i_1}, q_{i_2}, \dots)$ belong to the same switch, that is, cycles are "local" within switches.
Theorem 4 can be extended to:

***Theorem** 5:* Under the same first assumption of Theorem 4, but with a rephrased second assumption:

2. every switch implements a scheduling policy such that the logical-queues dependency graph is semi-acyclic;

then the network of switches achieves 100% throughput.
The proof of this theorem is given in Appendix V.

The logical-queues dependency graph can be made semi-acyclic according to the following *per-hop partition*. We partition the logical queues in Δ sets $Q^{(i)}$, with $0 \leq i < \Delta$, where Δ is the maximum number of hops traversed by a packet, i.e., the diameter of the network topology. $Q^{(0)}$ contains all the ingress logical queues; $Q^{(1)}$ contains the queues that follow some queue in $Q^{(0)}$ along the packets path; in general $Q^{(i)}$ comprises all the logical queues storing packets at the i-th hop in the network. The scheduling policy at each switch gives strict priority to queues in $Q^{(0)}$ with respect to $Q^{(1)}$, etc. Furthermore, we assume per-flow queueing: each logical queue can be accessed directly from the scheduler, and a packet can be enqueued/dequeued to/from a particular logical queue.

Note that the information on the distance expressed in number of hops from the network ingress point can be obtained either from the source address if a topological knowledge of the network is available at the router, or by maintaining a hop count in packet headers (for example using the time-to-live field of the IP packet headers).

Thanks to Theorem 5 we can state the following:

***Corollary** 4:* Consider a network of CIOQ switches with speedup 2. P-mSM with per-hop partition and per-flow queueing achieves 100% throughput.
The complexity of this algorithm is proportional to the number of maximum hops inside the network.

Note that, from a practical point of view, it is in general difficult to evaluate the real complexity of the considered algorithms, which is very dependent on implementation details. While we assumed a sequential complexity of $O(P^3)$ for MWM and of $O(P^2)$ for mSM, very efficient hardware implementations have been proposed to compute or approximate mSM (e.g, WFA [18] and iSLIP [3]).

A possible way to *approximate* a scheduling algorithm satisfying the assumptions of Theorem 4 consists of running a heuristic approximation of MWM with weights corresponding to logical queue sizes multiplied by coefficients large enough to mimic the strict-priority scheduling mechanisms that are required to break cycles in dependency graphs.

A final observation regards the fairness of hierarchical schedulers with priorities based on packet routing. We do not deal with this problem in this paper, but we recognize that priority systems can introduce relevant fairness problems, in particular when some flows are misbehaved and the traffic is not admissible.

V. Simulation Results

We consider the network of 8 IQ/CIOQ switches depicted in Fig. 1, in which continuous lines represent links between switches, and dashed lines represent information flows and their routing in the network; this network closely resembles the one studied in [8] to highlight instabilities in networks of IQ switches.

Note that each pair of switches (all pairs are alike) is traversed by a locally originated flow, a locally terminating flow, and an in-transit flow. The cell arrival process at the source of each flow is Bernoulli, and the normalized arrival rate for each flow is 0.33.

We compare the performance of the following 4 schemes: MWM with weights proportional to the queue lengths, A-OCF, P-mSM with per hop partition and P-mSM with per flow partition.

Figure 2 is very similar to graphs presented in [8] and [9], and shows that queue lengths take a divergent oscillating behavior when a local MWM scheduling is adopted. Queue lengths cannot be bounded. Figure 2 plots the curves only for the two switches IQS1 and IQS3, since IQS5 and IQS7 show almost identical behavior. Queues of IQS2, IQS4, IQS6 and IQS8 remain very small. The stationary network utilization for MWM is about 75%, corresponding to a normalized service rate 0.25 for each flow.

On the contrary, A-OCF and P-mSM with flow/hop partition show non-divergent behavior of the queues, whose occupancy remains small and finite; Figure 3 shows the occupancy of physical queues in IQS1 and IQS3 for A-OCF (very similar queues occupancy is experienced by both versions of P-mSM). All the three algorithms achieve the maximum throughput achievable in the network, i.e. 100% of network utilization. These results corroborate the results of theorem 3 and corollaries 3 and 4.

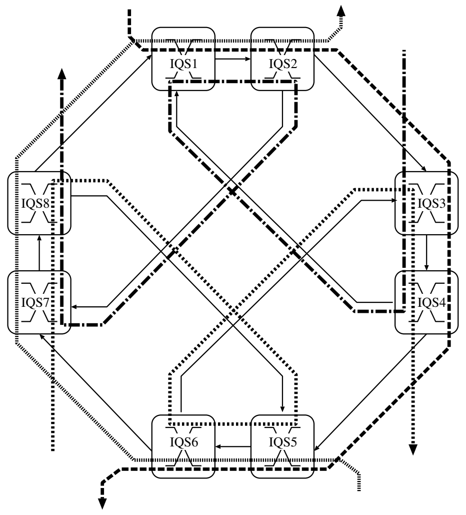

Fig. 1. The network of IQ switches considered in our simulation; continuous lines represent links between switches, and dashed lines represent information flows.

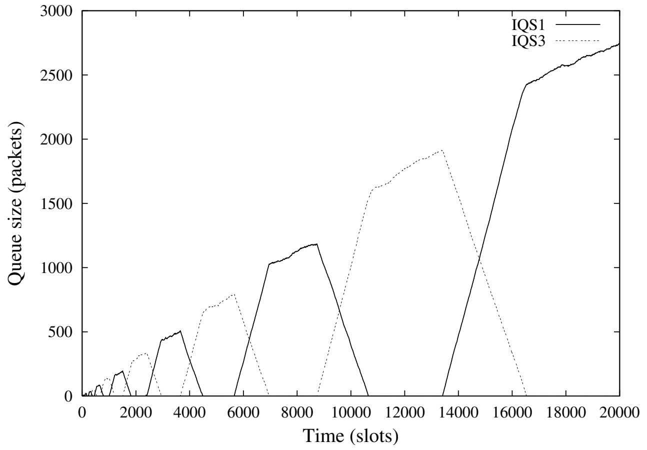

Fig. 2. Physical queues occupancy under MWM scheduler for the network of Figure 1. The two curves correspond to switches IQS1 and IQS3; IQS5 and IQS7 behave almost identically to IQS1 and IQS3. MWM achieves 75% of network utilization.

VI. Conclusions

This paper described scheduling policies that are stable in networks of IQ/CIOQ switches, i.e., they can achieve 100% throughput for any admissible traffic pattern satisfying the strong law of large numbers.

When the scheduling algorithm running at each switch can work only with state information which is local to the switch, we proposed a stable algorithm based on a Birkhoff-von Neumann decomposition. This algorithm is stable also if packet arrival rates are unknown and they are estimated on-line, using local information. We also proposed a stable algorithm based on Maximum Weight Matching, which is able to provide rate guarantees inside a network. A corollary of these findings is the extension of the results in [4] on the stability of the Oldest Cell First policy in an isolated IQ/CIOQ switch. Furthermore, two conjectures are reported which could

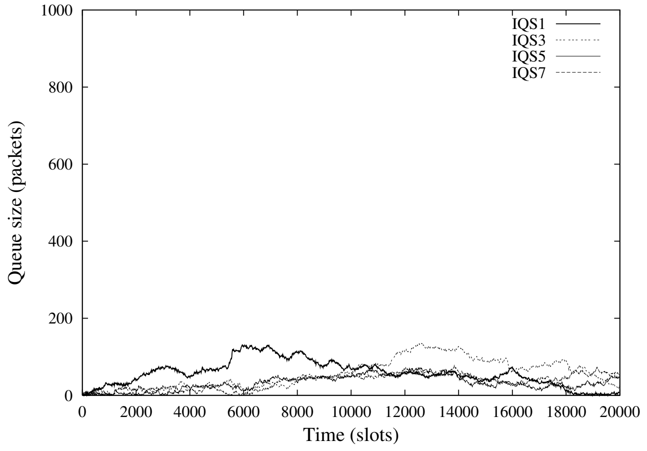

Fig. 3. Physical queues occupancy under A-OCF scheduler for the network of Figure 1. The curves correspond to the 4 switches IQS1, IQS3, IQS5 and IQS7. Behavior of P-mSM with both partitions is very similar to A-OCF. All these algorithms achieve 100% of network utilization.

be an interesting research field for future work.

If the scheduling algorithm can have some limited information about the routing of packet flows, we showed that strict-priority schedulers, based on maximal size matching algorithms, are stable when speedup 2 is allowed within the switching fabric.

Although all these results have mainly a theoretical relevance, they can be seen as a solid base to develop practical algorithms that exhibit reliable throughput properties in very large networks.

References

[1] T.Anderson, S.Owicki, J.Saxe, C.Thacker, "High Speed Switch Scheduling for Local Area Networks", *ACM Transactions on Computer Systems*, Nov.1993, pp. 319-352.

[2] N.McKeown, *Scheduling Algorithms for Input-Queued Cell Switches*, Ph.D. Thesis, Un. of California at Berkeley, 1995.

[3] N.McKeown, A.Mekkittikul, V.Anantharam, J.Walrand, "Achieving 100% Throughput in an Input-Queued Switch", *IEEE Transactions on Communications*, vol.47, n.8, Aug.1999, pp. 1260-1272.

[4] J.G.Dai, B.Prabhakar, "The Throughput of Data Switches with and without Speedup", *IEEE INFOCOM 2000*, Tel Aviv, Israel, Mar.2000, pp.556-564.

[5] N.McKeown, A.Mekkittikul, "A Pratical Scheduling Algorithm to Achieve 100% Throughput in Input-Queued Switches", *IEEE INFOCOM 98*, San Francisco, CA, USA, Apr.1998, pp.792-799.

[6] S.T.Chuang, A.Goel, N.McKeown, B.Prabhakar, "Matching Output Queuing with Combined Input and Output Queuing", *IEEE Journal on Selected Areas in Communications,* vol.17, n.6, Dec.1999, pp.1030-1039.

[7] E.Leonardi, M.Mellia, F.Neri, M.Ajmone Marsan, "On the Stability of Input-Queued Switches with Speedup", *IEEE/ACM Transactions on Networking,* vol.9, n.1, Feb.2001, pp.104-118.

[8] M.Andrews, L.Zhang, "Achieving Stability in Networks of Input-Queued Switches", *INFOCOM 2001*, Anchorage, Alaska, Apr.2001, pp.1673-1679.

[9] M.Ajmone Marsan, E.Leonardi, M.Mellia, F.Neri, "On the Throughput Achievable by Isolated and Interconnected Input-Queued Switches under Multicass Traffic", *INFOCOM 2002,* New York, NY (USA), June 2002.

[10] J.G.Dai, "Stability of Fluid and Stochastic Processing Networks", Miscellanea Publication n.9, Centre for Mathematical Physichs and Stochastic, Denmark (http://www.maphysto.dk), Jan.1999.

[11] J.G.Dai, "On Positive Harris Recurrence of Multiclass Queueing Networks: a Unified Approach Via Fluid Limit Models", *Annals of Applied Probability*, n.5, pp.49-77, 1995.

[12] F.Baskett, K.M.Chandy, R.R.Muntz, F.Palacios, "Open, Closed and Mixed Networks with Different Classes of Customers", *Journal of the ACM,* vol.22, n.2, April 1975, pp.248-260.

[13] M.Bramson, "Convergence to Equilibrium for Fluid Models of FIFO Queueing Networks", *Queueing Systems*, vol.22, 1996, pp.5-45.

[14] C.-S.Chang, W.-J.Chen, H.-Yi Huang, "Birkhoff-von Neumann Input Buffered Crossbar Switches", *INFOCOM 2000*, Tel Aviv, Israel, Apr.2000, pp.1614-1623.

[15] M.Katevenis, S.Sidiropoulos, C.Courcoubetis, "Weighted Round-Robin Cell Multiplexing in a General-Purpose ATM Switch Chip", *IEEE Journal on Selected Areas in Communications,* vol.9, n.8 , Oct.1991, pp.1265-1279.

[16] J.Y. Le Boudec, P. Thiran, "Network Calculus: A Theory of Deterministic Queuing Systems for the Internet", *Springer Publishing Company*, Jul.2001.

[17] F.P. Kelly, *Reversibility and Stochastic Networks,* John Wiley, New York, 1979.

[18] Y. Tamir, H.-C. Chi, "Symmetric Crossbar Arbiters for VLSI Communication Switches", *IEEE Transactions on Parallel and Distributed Systems*, vol.4, n.1, Jan.1993, pp.13-27.

Appendix I
Proof of Theorem 1

Proof: To prove the theorem we apply the fluid model methodology. In [4] the deterministic fluid model of a switch operating under some matching algorithm has been formally described. We consider an extension of that fluid model to a network of switches. Let $\Pi = \{\pi\}$ be the set of all possible network-wide matchings (which can be viewed as the composition of S single-switch matchings). The fluid equations of the system, corresponding to (2), are:

$$X(t) = X(0) + \Lambda t - D(t)(I-R) \tag{6}$$

$$D(t) = \sum_{\pi\in\Pi} \pi T_\pi(t) \tag{7}$$

$$\sum_{\pi\in\Pi} T_\pi(t) = t \tag{8}$$

where $X(t) \geq 0$, and $T_\pi(t)$ is a non-decreasing function giving the cumulative amount of time that matching π has been used up to time t. Since $W = \Lambda(I-R)^{-1}$,

$$\dot{X}(t) = (W - \dot{D}(t))(I-R)$$

where:

$$\dot{d}^{(k)}(t) > w^{(k)} + \epsilon \qquad \text{whenever} \quad x^{(k)}(t) > 0 \tag{9}$$

$$\dot{d}^{(k)}(t) = \dot{e}^{(k)}(t) \geq w^{(k)} \quad \text{whenever} \quad x^{(k)}(t) = 0 \tag{10}$$

for some $\epsilon > 0$, due to the W-compliance of the scheduling policy. Note that in (10) the first equality was proved in [10] for a general system of queues, whereas the inequality can be shown as follows: consider the case when all the progenitor queues of $q^{(k)}$ (i.e., queues that precede $q^{(k)}$ on the same packet flow) are empty, since the equality in (10) holds for every progenitor, then $\dot{e}^{(k)}(t) = w^{(k)}$. If there are some non empty progenitor queues then considering the last non empty progenitor h, and applying (9) it results: $\dot{e}^{(k)}(t) = \dot{d}^{(h)}(t) > w^{(h)} = w^{(k)}$.

Now consider the Lyapunov function $\mathcal{L}(X(t)) = X(I-R)^{-1}\mathbb{1}^T$, where $\mathbb{1}$ is a vector in $\mathbb{R}^N$ whose components are all equal to 1. Rate stability is guaranteed by a negative drift of Lyapunov function; in other words, if $\dot{\mathcal{L}}(X(t)) \leq 0$ when $\mathcal{L}(X(t)) > 0$, then $X(t) = 0$ is the only solution of the fluid model with initial condition $X(0) = 0$ and the system of queues is rate stable.

For every regular point t such that $\|X(t)\| > 0$, we can estimate the drift of the Lyapunov function:

$$\dot{\mathcal{L}}(X(t)) = \dot{X}(t)(I - R)^{-1} 1\!\!I^T = (W - \dot{D}(t)) 1\!\!I^T =$$
$$= \sum_{k:x^{(k)}(t)>0} [w^{(k)} - \dot{d}^{(k)}(t)] + \sum_{k:x^{(k)}=0} [w^{(k)} - \dot{d}^{(k)}(t)] \leq$$
$$\leq - \sum_{k:x^{(k)}(t)>0} \epsilon \leq -\epsilon \quad (11)$$

From the latter we can conclude that $X(t) = 0$ is the only solution of the fluid model equations under the initial condition $X(0) = 0$. Thus the system of queues is rate stable.

Similar proof can be devised to show that also any W-weak-compliant policy is rate stable. ■

Note that the proof of this theorem strongly resembles the proof of Theorem 2.4.9 reported in [10], of which it may be considered an extension to networks of IQ/CIOQ switches.

APPENDIX II
PROOF OF THEOREM 2

Proof: First we prove the initial part of the theorem, regarding the BvN decomposition. Let $\rho = ||W||_{IO}$. Consider a sample path of the system evolution for which $\hat{W}_n \to W$. Since the scheduler follows a BvN approach driven by an on-line estimation $\hat{W}_n$, the service rate assured at each queue is:

$$r_n^{(k)} \geq \frac{\hat{w}_n^{(k)}}{||\hat{W}_n||_{IO}}$$

Since, by assumption, $\hat{W}_n \to W$ w.p.1, then, given a queue $q^{(k)}$ such that $w^{(k)} > 0$ and any arbitrary positive ϵ, there exists a t_0 such that, for any $m > t_0$:

$$w^{(k)} - \epsilon < \hat{w}_m^{(k)} < w^{(k)} + \epsilon$$

Thus, $||\hat{W}_m||_{IO} \leq \rho + N\epsilon$, and:

$$r_m^{(k)} \geq \frac{\hat{w}_m^{(k)}}{||\hat{W}_m||_{IO}} \geq \frac{w^{(k)} - \epsilon}{\rho + N\epsilon}$$

Condition:

$$r_m^{(k)} \geq w^{(k)} \quad (12)$$

is satisfied when:

$$w^{(k)} - \epsilon \geq w^{(k)}\rho + w^{(k)}N\epsilon \Rightarrow \epsilon \leq \frac{w^{(k)}(1-\rho)}{1 + Nw^{(k)}}$$

If we define:

$$w_{max} = \max_k\{w^{(k)}\} \qquad w_{min} = \min_k\{w^{(k)} : w^{(k)} > 0\}$$

we can choose

$$\epsilon = \frac{w_{min}(1-\rho)}{1 + Nw_{max}}$$

to satisfy (12) and guarantee that the policy is (t_0, W)-compliant, hence achieving 100% throughput. Since the fluid limits of the system are not affected by the behavior of the scheduling policy in a transient phase of finite length, the previous results can be immediately generalized to W-compliant policies.

Now we have to prove the second part of the theorem, i.e., to show that:

$$\hat{w}_n^{(k)} = \frac{1}{n}\sum_{m=0}^{n-1} e_m^{(k)} \to w^{(k)} \qquad \text{w.p.1.}$$

Since we assume deterministic per-flow routing, it is possible to partition the set of queues $\{q^{(k)}\}$ into different sets $Q^{(h)}$, according to the following rule: $Q^{(i)}$ contains those logical queues storing packets that have already traversed i switches in the network. Thus $Q^{(0)}$ contains the queues of packets that just entered the network, while $Q^{(1)}$ contains the logical queues that follows those in $Q^{(0)}$ along flow routes.

Since the arrival process satisfies the SLLN,

$$\hat{w}_n^{(k)} = \frac{1}{n}\sum_{m=0}^{n-1} e_m^{(k)} \to w^{(k)} \qquad k \in Q^{(0)} \qquad \text{w.p.1}$$

thus queues in $Q^{(0)}$ are rate stable. Thanks to the results shown in the first part of this proof, there exists t_0 such that, for $m \geq t_0$, $r_m^{(k)} \geq w^{(k)}$ and:

$$\frac{1}{n}\sum_{m=0}^{n-1} e_m^{(k)} - d_m^{(k)} \to 0 \qquad k \in Q^{(0)} \qquad \text{w.p.1}$$

which implies: $1/n \sum_{m=0}^{n-1} d_m^{(k)} \to w^{(k)}$, w.p.1 i.e., the departure processes from queues in $Q^{(0)}$ satisfy the SLLN. But departure processes from queues in $Q^{(0)}$ are the arrival processes of queues in $Q^{(1)}$, then the arrival processes to queues $Q^{(1)}$ satisfy the SLLN. As a consequence also queues in $Q^{(1)}$ are rate stable. In general, if we assume that queues in $Q^{(n)}$ are rate stable, then the departure processes from queues in $Q^{(n)}$ satisfy the SLLN. But as departure processes from queues in $Q^{(n)}$ are the arrival processes for queues in $Q^{(n+1)}$, then the arrival processes to queues $Q^{(n+1)}$ satisfy the SLLN. As a consequence, also queues in $Q^{(n+1)}$ are rate stable. Then, by mathematical induction, every queue in the network is rate stable. ■

APPENDIX III
PROOF OF THEOREM 3

Proof: Consider again the fluid equations (6), (7) and (8). Since the scheduler realizes a MWM, it is useful to define:

$$\Pi'(t) = \{\pi' : \langle \pi', \Phi(t) \rangle = \max_{\pi} \langle \pi, \Phi(t) \rangle\} \quad (13)$$

where $\langle v_1, v_2 \rangle = v_1 v_2^T$ is the scalar product between the two vectors v_1 and v_2; the vector $\Phi(t) = [\phi^{(k)}]$ collects the weights of all the logical queues. Hence the fluid equations are:

$$X(t) = X(0) + \Lambda t - D(t)(I - R) \tag{14}$$

$$\dot{D}(t) = \sum_{\pi \in \Pi'(t)} \pi \dot{T}_\pi(t) \tag{15}$$

$$\sum_{\pi \in \Pi'(t)} T_\pi(t) = t \tag{16}$$

$$\phi^{(k)}(t) = t - \frac{\bar{d}^{(k)}(t)}{w^{(k)}} \tag{17}$$

being $\bar{d}^{(k)}(t) = \sum_{n \leq t} d^{(k)}(n)$ the cumulative number of services at queue k up to time t, whose fluid limit is $D(t)$. Note that $\bar{d}^{(k)}(t) \to \infty$ for $t \to \infty$. Equation (17) is obtained by observing that, for $n \to \infty$, $f(n) \to n/w^{(k)}$ and, for $t \to \infty$,

$$f(\bar{d}^{(k)}(t)) \to \frac{\bar{d}^{(k)}(t)}{w^{(k)}}$$

Note that (17) estimates the average waiting time of HOL cells.

Let $\Gamma = [\gamma^{(i,j)}]$ be the diagonal matrix with $\gamma^{(k,k)} = w^{(k)}$, and let Γ^{-1} be inverse of Γ. From (17) we have:

$$\dot{\Phi}(t) = 1\!\!I - \dot{D}(t)\Gamma^{-1} \tag{18}$$

Consider the Lyapunov function:

$$V(\Phi(t)) = \frac{1}{2}\langle \Phi(t)\Gamma, \Phi(t)\rangle \tag{19}$$

Hence, to prove stability, we want to show that:

$$\frac{d}{dt} V(\Phi(t)) \leq 0 \tag{20}$$

Indeed we have:

$$\begin{aligned}\frac{d}{dt} V(\Phi(t)) &= \frac{1}{2}\langle \dot{\Phi}(t)\Gamma, \Phi(t)\rangle + \frac{1}{2}\langle \Phi(t)\Gamma, \dot{\Phi}(t)\rangle = \\ &= \langle \dot{\Phi}(t)\Gamma, \Phi(t)\rangle = \langle [1\!\!I - \dot{D}(t)\Gamma^{-1}]\Gamma, \Phi(t)\rangle = \\ &= \langle \Gamma, \Phi(t)\rangle - \langle \dot{D}(t), \Phi(t)\rangle = \\ &= \langle \Gamma, \Phi(t)\rangle - \langle \sum_{\pi \in \Pi'(t)} \pi \dot{T}_\pi(t), \Phi(t)\rangle = \\ &= \langle \Gamma, \Phi(t)\rangle - \sum_{\pi \in \Pi'(t)} \dot{T}_\pi(t)\langle \pi, \Phi(t)\rangle \leq 0 \end{aligned} \tag{21}$$

The last inequality holds because: (i) $\langle \pi, \Phi(t)\rangle$ is the weight of the MWM, which depends only on t by (13), (ii) from (16) it is $\sum_{\pi \in \Pi'(t)} \dot{T}_\pi(t) = 1$, and (iii) Γ belongs to the convex hull generated by Π.

Thanks to (21), $\Phi(0) = 0$ implies $\Phi(t) = 0$. Hence, from (17), we can say:

$$\frac{\bar{d}^{(k)}(t)}{w^{(k)}} = t$$

$$\lim_{t \to \infty} \frac{\bar{d}^{(k)}(t)}{t} = w^{(k)} \qquad \text{w.p.1}$$

$$\lim_{t \to \infty} \frac{D(t)}{t} = W$$

which corresponds to the rate stability conditions for $X(t)$. ■

APPENDIX IV
PROOF OF THEOREM 4

Proof: This proof is again obtained by means of fluid models of the network of IQ/CIOQ switches. From (14):

$$\dot{X}(t) = \Lambda - \dot{D}[X](t)(I - R)$$

where the notation $D[X](t)$ explicitly expresses the fact that $D(t)$ depends on $X(t)$. In addition:

$$||D[X](t)||_{IO} \leq 1$$

The policy is locally stable, then the functional relation between $D(t)$ and $X(t)$ is such that, for any admissible Γ with $||\Gamma||_{IO} < 1$, equation:

$$\dot{X}(t) = \Gamma - \dot{D}[X](t) \tag{22}$$

admits $\dot{X}(t) = 0$ as the only possible solution with initial condition $X(0) = 0$.

Let us now establish the following partial ordering rule among logical queues in the network. Let $q^{(i)}$ and $q^{(j)}$ be two queues; we say that $q^{(i)} < q^{(j)}$ (i.e $q^{(i)}$ is an "ancestor" of $q^{(j)}$) if there exists a directed path on the dependency graph from $q^{(i)}$ to $q^{(j)}$. Since the graph is acyclic, it can be easily verified that "$<$" defines a good partial ordering relation.

Now we partition the queues in the network according to the following rule (note that this partition is different from the one used in the proof of Theorem 2): Let $V^{(k)}$ be the k-th partition of the network of queues. $V^{(0)}$ comprises the queues that do not have any ancestor according to the ordering relation defines above. $V^{(1)}$ contains those queues that have all ancestors in $V^{(0)}$; $V^{(2)}$ comprises those queues that have all ancestors in $V^{(0)}$ or $V^{(1)}$, etc. Note that, by definition, in any set $V^{(i)}$ there cannot be two logical queues that refer to the same flow.

Consider now a rate stable policy which is applied *only* on queues in $V^{(0)}$. Queues in $V^{(0)}$ are surely stable since:

- Queues in $V^{(0)}$ are fed by network ingress traffic, that thus satisfy the SLLN; the fluid arrival process to queue $q^{(k)} \in V^{(0)}$ is a constant-rate process with parameter $w^{(k)}$.
- No queues in $V^{(i)}$ with $i > 0$ can prevent queues in $V^{(0)}$ by obtaining service; thus the scheduling policy ignores queues in $V^{(i)}$ with $i > 0$ when it allocates service to queue in $V^{(0)}$.
- If the queues in $V^{(i)}$ with $i > 0$ are ignored, the set of queues $V^{(0)}$ corresponds to a system of isolated IQ/CIOQ switches subject to an admissible process satisfying the SLLN. Then the fluid evolution for queues in $V^{(0)}$ is described by:

$$\dot{X}^{(0)}(t) = W^{(0)} - \dot{D}^{(0)}[X^{(0)}](t) \tag{23}$$

$$||D^{(0)}[X^{(0)}](t)||_{IO} \leq 1 \tag{24}$$

where components of $X^{(0)}$, $W^{(0)}$ and $D^{(0)}[X^{(0)}]$, corresponding to queues in $V^{(0)}$, are equal to those of vectors $X(t)$, W and $D[X](t)$ respectively, while the other components are null.

Note that Equation (23) is structurally identical to the fluid equation (22); then the only solution corresponding to the null initial condition is $X^{(0)}(t) = 0$. All queues in $V^{(0)}$ are rate stable and the departure process from queues in $V^{(0)}$ must satisfy the SLLN.

Now, let us focus on queues in $V^{(1)}$; they are fed by other ingress processes or departure processes of queues in $V^{(0)}$. Thus the ingress processes to queue $V^{(1)}$ satisfy the SLLN. Let us consider the fluid model of $V^{(0)}$ and $V^{(1)}$; we know that queues in $V^{(0)}$ are rate stable, thus $x^{(k)}(t) = 0$ is the only solution of fluid model equations with null initial condition for every queue $q^{(k)} \in V^{(0)}$. As a consequence $\dot{d}^{(k)}(t) = w^{(k)}$ for every queue in $V^{(0)}$ and the departure process from queues in $V^{(0)}$ satisfy the SLLN.

Thus, the fluid evolution of queues in $V^{(0)}$ and $V^{(1)}$ can be described by:

$$\dot{X}^{(1)}(t) = W^{(1)} - \dot{D}^{(1)}[X^{(1)}](t) \qquad (25)$$

$$||D^{(1)}[X^{(1)}](t)||_{IO} \leq 1 \qquad (26)$$

where components of $X^{(1)}$, $W^{(1)}$ and $D^{(1)}[X^{(1)}]$, corresponding to queues in $V^{(0)}$ and $V^{(1)}$ are equal to those of vectors $X(t)$, W and $D[X](t)$ respectively, while the other components are null.

Then the departure processes from queues in $V^{(1)}$ must satisfy the SLLN.

By iterating the same argument on all partitions $V^{(i)}$ with $i = 2, 3, \ldots$ in sequence, we obtain that all the queues of the network are rate stable. ■

APPENDIX V
PROOF OF THEOREM 5

Proof: The proof of this theorem is carried out in a way very similar to the proof of Theorem 4. All the major conceptual steps are repeated.

The only significant difference resides in the fact that, since the dependency graph contains cycles, we cannot establish the same partial order relation among logical queues as before. Thus we have first to divide all the logical queues in equivalence classes, putting in the same class all the logical queues that belong to the same cycle. Note that queues belonging to the same equivalence class are, by construction, located at the same switch, and thus refer to different flows.

Now, neglecting in the dependency graph edges among queues in the same equivalence class, i.e., imploding in a single node all the nodes of the dependency graph corresponding to queues inside the same equivalence class, we can proceed as in the proof of Theorem 4, defining sets $V^{(k)}$ of equivalence classes of logical queues. Also in this case set $V^{(k)}$ cannot contain two queues that belong to the same flow. Moreover queues in $V^{(0)}$ are fed by ingress traffic, while queues in $V^{(i)}$ with $i > 0$ are fed by packets leaving queues in $V^{(i-1)}$. Thus the same arguments as before apply, and the assert is proved. ■