# Distributed scheduling policies in networks of input-queued packet switches

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

*Abstract*— **Scheduling algorithms for input-queued packet switches have been widely researched. It has been shown that various classes of scheduling algorithms provide guarantees on stability and on average delay for single switches. However, recent research has demonstrated that most of these scheduling algorithms do not guarantee stability for networks of switches.**

**Most of the research that treats networks of switches proposes switching policies that require coordination among the single switches. The problem to find *distributed* scheduling policies that guarantee the stability of a network of switches has so far only been investigated for a policy based on a computationally very complex maximum weight matching algorithm.**

**In this paper, we investigate *distributed* scheduling algorithms of *low complexity* that belong to the classes of maximal weight matching algorithms, $p$-maximal weight matching algorithms for switch architectures based on a space-division multiplexing extension, and MNCM algorithms. For these scheduling algorithms, we prove the stability of networks of input-queued switches where all switches deploy the same scheduling policy. We also show that networks of input-queued switches in which specific classes of different switching policies are deployed simultaneously at different switches are stable.**

## I. Introduction and Motivation

The increase in internet traffic and the progress in optical transmission technologies create a need for fast switching technologies in the internet core. Research on switch architectures and scheduling algorithms has mostly focused on input-queued (IQ) and combined input output-queued (CIOQ) switches. A typical CIOQ switch is shown in figure 1. To avoid head-of-line blocking, a typical $NxN$ CIOQ switch has $N$ virtual output queues $VOQ_{i,j}$, $1 \leq j \leq N$ at each input $i$. Packets that arrive at input $i$ and are destined for output $j$ and not forwarded immediately upon their arrival, are buffered in $VOQ_{i,j}$. The switching core works with a speedup of $S, S \geq 1$, i.e., it works at a speed $S$ times faster than the speed of the input and the output links. If $S > 1$, packets are also buffered at the outputs. The switching core is typically modeled as a crossbar, such that not more than one packet can be sent simultaneously from the same input or to the same output.

The choice of the scheduling algorithm is a major design criteria for switches. The problem of finding an optimal switch configuration for a specific switch state can be modeled as the problem of finding a maximum weight matching of a bipartite $N \times N$ graph. In [12] and [14], it has been shown that for a speedup of $S = 1$, a maximum weight matching algorithm can guarantee the stability of a single switch if the weights are chosen proportionally to the lengths of the VOQs. However, the implementation

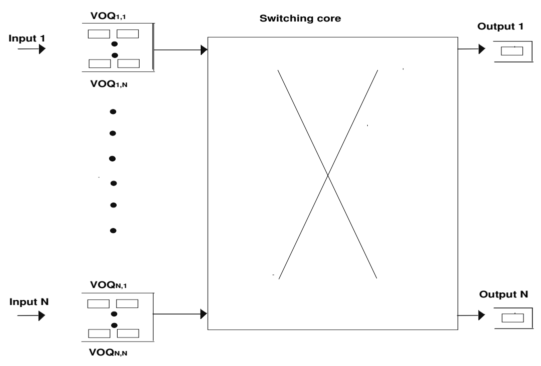

Fig. 1. Architecture of an input-queued switch

of maximum weight algorithms is impractical as they require the solution of an optimization problem that is based on the Hungarian method and has a known complexity of $O(N^3)$ (see [17]).

Therefore, various kinds of computationally less complex scheduling algorithms have been investigated. In particular, the class of maximal weight matching algorithms has been widely investigated ([6], [10], [11], [15]). It has been shown in [7] that under the assumption of admissible traffic, every maximal weight matching algorithm deployed with a speedup of 2 guarantees the stability of the switch. In an improvement on the work in [7], it was shown in [4] and [5] that a switch that deploys any maximal weight matching algorithm is stable even with a speedup $S$ that is slightly smaller than 2.

However, in view of the steady increase in optical transmission speed, even a moderate speedup of 2 or less - as required for the stability of maximal weight matching algorithms - becomes increasingly difficult to implement. To overcome this difficulty, in [18], a CIOQ switch with a space-division multiplexing expansion has been proposed. In contrast to the crossbar model of a switch that assumes the existence of only one connection between each input and output pair, this model allows more than one connection per input - output pair. It is shown in [18] that a switch that provides two connections between each input and output pair and deploys a 2-maximal weight matching algorithm with a speedup of $S = 1$ guarantees 100% throughput of the switch.

In [16], a new class of maximal weight matching algorithms called the class of maximum node containing matching (MNCM) algorithms, is introduced. It is shown in

[16] that MNCM algorithms provide 100% throughput for a switch with a speedup of $S = 1$. The gain in speedup compared to the traditional maximal weight matching algorithms discussed above comes at the cost of a higher computational complexity. Whereas traditional maximal weight matching algorithms have a complexity of $O(N^2)$, the MNCM algorithms are based on the solution of a maximum size problem which has a complexity of $O(N^{2.5})$.

It was shown in [3] that in a specific network of input-queued switches that is not overloaded, a maximum weight matching policy as defined in [12] that is implemented independently at each switch does not guarantee stability. The authors give a specific example where under non-overloaded conditions the number of packets buffered at specific $VOQ$s at some switches in the network grows unlimited. This result showed that the previous work on maximum weight matching algorithms in isolated switches is of limited practical value in networks of switches. Following the ideas in [3], it can also be shown that networks of switches that deploy maximal weight matching, $p$-maximal weight matching and MNCM algorithms can become instable in non-overloaded networks.

In [1] and [3], switching policies that guarantee the stability of all switches within a network, but require coordination and cooperation among the switches within the network, are presented. 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.

Due to the impracticality of maximum weight matching algorithms described above, it is of interest to understand if *distributed* scheduling algorithms of *low complexity* that guarantee 100% throughput in a network of input-queued switches exist. This paper shows for the first time the existence of *distributed* switching policies of *low complexity* that are based on maximal weight matching algorithms, $p$-maximal weight matching algorithms and MNCM algorithms and that guarantee 100% throughput when all switches in the network deploy the same switching policy. In addition, we prove that a network of input-queued switches where each switch deploys any of these policies or a maximum weight matching policy as proposed in [2], also guarantees a 100% throughput. We use the theory of the Lyapunov function ([12], [9], [14]) and the fluid model methodologies ([7], [8]) to establish our results. Our proofs make use of new equations that describe the behavior of maximal weight matching and $p$-maximal weight matching algorithms.

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 define three types of scheduling policies and develop mathematical models to describe their behavior. For each policy, we present stability results for networks of input-queued switches. Finally, we show the stability of networks where different policies are deployed simultaneously at different switches. The proofs of the theorems are given in the appendix. 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 $q^k$ at the beginning of 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. We assume that packets arrive at a queue at the beginning of a timeslot and depart from a queue at the end of a timeslot. 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 $A_n = (a_n^1, ..., a_n^K)$ 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 from (1) and (2):

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

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

$$\lim_{n \to \infty} \frac{\sum_{i=1}^{n} A_i}{n} = \Lambda \qquad \text{w.p.1}, \tag{4}$$

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

We now calculate the average workload of the logical queues $q^k$ which we denote by $W = (w^1, ..., w^K)$. 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 $(I - R)^{-1} = I + R + R^2 + ...$, we find that the overall average workload at the logical queues $q^k$ denoted as 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 norm for a vector $Z \in \mathbb{R}^K$ :

**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:

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

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

$$||W||_{maxL} < 1. \tag{6}$$

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

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

## *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 (The concept of virtual output queues is explained in the introduction of this article). In this model we neglect the output queues of the switches because instability can only occur at the $VOQ$s (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 sections III $B$ - $F$, 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:

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

Because we assume a deterministic routing policy, the necessary condition for rate stability given in (6) 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:

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

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

## III. Stable local scheduling policies

In this section, we introduce three different local scheduling policies that guarantee 100% throughput in a network of IQ/CIOQ switches. We will initially present each scheduling policy in a non-distributed, centralized way, i.e. we will describe the scheduling algorithms as if it was in each timeslot executed by a central network server. The server calculates the configuration of all switches in the network by taking into account the actual state of each switch and then sends each switch its specific configuration. In section $F$, we will show how the individual scheduling policies can be implemented in a distributed fashion, i.e., in each timeslot, each switch calculates its own configuration taking into account only its own state.

### *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:

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

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

$$\phi^k_n \quad = \quad n - f_k(\overline{d}^k(n)) + C. \tag{9}$$

This choice of the weights is motivated by the fact that it will allow us to derive the relation (36), which in turn implies the rate stability (see definition 1) for the theorems 2, 4, 5 and 7 below. We set $\Phi_n = (\phi^1_n, .., \phi^K_n)$.

We see that for a fixed function $f(\cdot)$ that satisfies the relation (8), there exists a constant $C > 0$ such that $\forall n$, $n \geq 0$, there holds $f(\overline{d}^k_n) \leq \frac{\overline{d}^k_n}{w^k} - C$. 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 (8), one can always find a $C > 0$ such that the weights $\phi^k_n$ are positive $\forall n$, $n \geq 0$, $\forall k$, $1 \leq k \leq K$. This fact is important in view of the theorems 1 - 5 below, which all require the weights $\phi^k_n$ to be strictly 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 $\overline{a}^k(n) = \sum_{m\leq n} a^k_m$. 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 $\phi^k_n = n - [\overline{a}^k]^{-1}(p)$ denotes the age of the $p$-th packet at time $n$. At its departure time $n$, the age of the $p$-th packet is $n - [\overline{a}^k]^{-1}(\overline{d}^k_n)$.

### *B. Maximal weight matching algorithms*

In this section, we propose a maximal weight matching scheduling policy for a network of switches with a speedup of $S > H$.

For this purpose, we introduce some additional terminology. In this section, we call a timeslot as defined earlier an external timeslot defined as the time needed by a packet to arrive completely at an incoming link. As the switching core works at a speedup $S \geq 1$, an internal timeslot is defined as the time needed to transfer a packet through the switching core from an input to an output. Thus, the external timeslot from time $t$ to $t+1$ consists of the $S$ internal timeslots $[t + (k-1)/S, t + k/S]$, $1 \leq k \leq S$. For the sake of simplicity, we always assume that $S$ is an integer. The proofs of the theorems in this paper can be easily generalized for non-integer $S$. For any rational $S$, instead of considering the dynamics of the switch over one external timeslot as done in the proofs related to this chapter, the dynamics of the switch over $gS$ external timeslots such that $gS$ is an integer would have to be considered. If $S$ is non-rational, all proofs in this paper hold by replacing $S$ by a arbitrarily close, smaller rational number. We suppose that packets arrive at the beginning of an external timeslot $t$ and are transferred instantly at the end of an internal timeslot.

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

$$||D_n||_{IO} \leq S, \qquad \forall n \geq 1. \tag{10}$$

Further, for a given input $i$ and a given output $j$ at a given switch $b$, we define the set of all logical queues that either belong to the input $i$ or that are directed to the output $j$. We set $\forall b, i, j$, $1 \leq b \leq B$, $1 \leq i, j \leq N$,

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

For each set $\mathcal{S}_{b,i,j}$, we sum the average arrival rates for all logical queues that belong to $\mathcal{S}_{b,i,j}$, and define the maximum of these summations as $H$ :

$$H \quad = \quad \max_{\substack{1\leq b\leq B \\ 1\leq i,j\leq N}} \sum_{m\in\mathcal{S}_{b,i,j}} w^m. \tag{11}$$

From (7), we see $H < 2$.

We now give a formal definition of a maximal weight matching algorithm: For a set of positive weights $P^k$, $1 \leq k \leq K$, where $P^k$ is the weight assigned to the logical queue $q^k$, a maximal weight matching algorithm is defined 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.

We now develop an inequality that describes the dynamics of a maximal weight matching algorithm. This inequality will be used to prove the stability of a specific maximal weight matching scheduling policy in theorem 2. We define the weights of the algorithm as a function of the actual timeslot $n$ as follows: $P_n = (P_n^1, ..., P_n^K)$ are the weights at the beginning of the $n$-th external timeslot. $P_{n,s}^k$, $1 \leq s \leq S$, is the weight of the logical queue $q^k$ at the beginning of the $s$-th internal timeslot of the $n$-th external timeslot. Thus, $P_n = P_{n,1}, \forall n \geq 0$. We also define the departure vector of the $s$-th internal timeslot of the $n$-th external timeslot as $D_{n,s} = (d_{n,s}^1, ..., d_{n,s}^K)$ such that $D_n = \sum\limits_{s=1}^{S} D_{n,s}$, and

$$||D_{n,s}||_{IO} \leq 1, \tag{12}$$

$\forall n \geq 1$, $\forall 1 \leq s \leq S$. Now, we establish a lower bound for the weight of a matching calculated by a maximal weight matching algorithm during an external timeslot. In particular, we show that the weight of a matching calculated in an external timeslot is $1/H$-times larger than the sum of the products of the average arrival rate $w^k$ of a logical queue $q^k$ multiplied with the actual weight $P_{n,s}^k$ summed over all logical queues and all internal timeslot of the considered external timeslot.

**Theorem 1:** *For a network of IQ/CIOQ switches that applies a maximal weight matching algorithm for positive weights $P^k > 0$, $1 \leq k \leq K$, and a speedup-up of $S$, there holds for any timeslot $n$*

$$\frac{1}{H}\sum_{s=1}^{S}\sum_{k=1}^{K} P_{n,s}^k w^k \leq \sum_{k=1}^{K}\sum_{s=1}^{S} P_{n,s}^k d_{n,s}^k.$$

*Proof:* We first analyze a maximal weight matching algorithm in the first internal timeslot. In its first iteration, the algorithm selects the queue with the largest weight, say $P_n^{k_1}$ with $k_1 \in \mathcal{S}_{b_1,i_1,j_1}$, for transfer. Thus, using that $P^k > 0$, we see from (11):

$$\begin{aligned} P_n^{k_1} H &\geq P_n^{k_1} \sum_{m \in \mathcal{S}_{b_1,i_1,j_1}} w^m \\ &\geq \sum_{m \in \mathcal{S}_{b_1,i_1,j_1}} P_n^m w^m. \end{aligned}$$

All logical queues $q^k$ with $k \in \mathcal{S}_{b_1,i_1,j_1}$ are removed. In the second iteration, the remaining queue with the largest weight, say $P_n^{k_2}$ with $k_2 \in \mathcal{S}_{b_2,i_2,j_2}$ is chosen. Thus,

$$\begin{aligned} P_n^{k_2} H &\geq P_n^{k_2} \sum_{\substack{m \in \mathcal{S}_{b_2,i_2,j_2} \\ m \notin \mathcal{S}_{b_1,i_1,j_1}}} w^m \\ &\geq \sum_{\substack{m \in \mathcal{S}_{b_2,i_2,j_2} \\ m \notin \mathcal{S}_{b_1,i_1,j_1}}} P_n^m w^m. \end{aligned} \tag{13}$$

The matching algorithm stops after $h$, $h \leq BN$ iterations when none or only empty queues remain. For each of these $h$ iterations, an inequality analogous to (13) holds. If any empty queues remain, we hypothetically continue to run the algorithm $(BN - h)$ times and produce valid inequalities as in (13) where both sides are equal to zero. We note that $d_n^k = 1$ only for the logical queues $q_k$ chosen for transmission. Thus, summing over all $BN$ inequalities, we obtain

$$\sum_{m=1}^{BN} P_n^{k_m} = \sum_{m=1}^{BN} P_n^{k_m} d_n^k = \sum_{k=1}^{K} P_n^k d_n^k \geq \frac{1}{H}\sum_{k=1}^{K} P_n^k w^k. \tag{14}$$

Applying this analysis to all $S$ internal timeslots, we obtain from (14)

$$\sum_{s=1}^{S}\sum_{k=1}^{K} P_{n,s}^k d_{n,s}^k \geq \frac{1}{H}\sum_{s=1}^{S}\sum_{k=1}^{K} P_{n,s}^k w^k. \qquad \square$$

We state the next corollary as an immediate consequence of theorem 1:

**Corollary 1** *For any input buffered switch that applies a maximal weight matching algorithm and a speedup-up of $S$, if there holds for a positive constant $C_1$, $|P_{n,s}^k - P_{n,s+1}^k| \leq C_1$ $\forall k, n, s$, $1 \leq k \leq K$, $n \geq 1$, $1 \leq s \leq S-1$, then there holds for any timeslot $n$ :*

$$\frac{S}{H}\sum_{k=1}^{K} P_n^k(t) w^k \leq \sum_{k=1}^{K} P_n^k d_n^k + \frac{S(S-1)}{2} K C_1 \left(1 + \frac{1}{H}\right).$$

*Proof:* The corollary is derived from theorem 1 by approximating the $P_{n,s}^k$ by $P_n^k$ - using the assumptions of the corollary - and applying the estimates $|w^k| < 1$, $|d_n^k| \leq 1$ which follow from (7) and (12) respectively.
The assumption of corollary 1 that weights only change by a finite, bounded amount between two iterations of the algorithm is valid for most weight functions proposed in the literature. In particular, it applies to the weight function defined in (9).

We now state the main result of this section. The following theorem states the existence of a class of local scheduling policies that are based on a maximal weight matching algorithm using a speedup larger than $H$ and that achieve 100% throughput in a network of IQ/CIOQ switches.

**Theorem 2:** *A network of IQ/CIOQ switches that implements a maximal weight matching scheduling policy with a speedup of $S > H$ in which the weight $\phi_n^k$ of queue $q^k$ at time $n$ is defined as in (9) and $\phi_n^k > 0$, achieves 100% throughput.*
*Proof:* The proof is given in the appendix.

### C. p-maximal weight matching algorithms

The application of $p$-maximal weight matching algorithms to the problem of determining the configuration of a packet switch was first described in [18]. In contrast to earlier switch architectures that allow only one packet to be sent from one input or to one output in each timeslot, in [18] a switch with a space-division multiplexing expansion is presented that deploys $p$, $p \geq 2$ connections between each input and output pair.

The scheduling problem is modeled as a $p$-matching problem on a bipartite graph whose nodes are the inputs and outputs of a switch and the edges are the connections between inputs and outputs. A $p$-matching is defined as a matching, such that no node of the matching is incident with more than $p$ nodes. Thus, for a $p$-maximal weight matching algorithm the following relation holds:

$$||D_n||_{IO} \leq p. \tag{15}$$

In this paper, we consider a specific $p$-maximal weight matching algorithm that can be considered as a generalization of the maximal weight matching algorithm discussed in the previous section. The algorithm is executed at the beginning of each timeslot. During each execution, it successively schedules up to $pBN$ transmissions from the logical queues at the inputs of the switches to the destination outputs. We introduce the additional variables $c_r$, $0 \leq r \leq 2NB$, where $r$ indexes all inputs and outputs in the considered network and $2NB$ is the total number of all inputs and outputs in the network. During the execution of the algorithm, if $r$ indexes an input, the value of the variable $c_r$ expresses the actual number of times that a cell has been scheduled to be sent from the input $r$ to any output. If $r$ indexes an output, $c_r$ equals the actual number of times a cell has been scheduled to be sent to the output $r$ from any input. At the start of the algorithm, the variables $c_r$ are initialized equal to zero and they can take integer values between 0 and $p$.

We now define a $p$-maximal weight matching algorithm: With regard to a set of positive weights $P^k$, $1 \leq k \leq K$, where $P^k$ is the weight assigned to the logical queue $q^k$, a $p$-maximal weight matching algorithm is defined as follows:

1. Initially, all logical queues $q^k$, $1 \leq k \leq K$, are considered potential choices for a cell transfer. All $c_r$, $1 \leq r \leq 2NB$ are set equal to 0.
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. $c_{i_1}$ and $c_{j_1}$ are updated as $c_{i_1} = c_{i_1} + 1$ and $c_{j_1} = c_{j_1} + 1$. If $c_{i_1} = p$, all logical queues $q^k$ with $k \in Q_I(b_1, i_1)$ are removed. If $c_{j_1} = p$, all logical queues $q^k$ with $k \in Q_O(b_1, j_1)$ are removed.
4. If all $q^k$ are removed, the algorithm terminates. Else go to step 2.

The weights of the logical queues at the beginning of the $n$-th timeslot have been defined as $P_n^k$. After each iteration of the algorithm, some weights are updated. We define $P_n^{k,*}$ as the actual weight of the logical queue $q^k$ between inclusively the first and the large iteration of the algorithm. At the beginning of the first iteration we initialize $P_n^{k,*} = P_n^k$. As the weight of a queue $q^k$ is updated at most $p$ times during the execution of the $p-$maximal weight matching algorithm, the value of $P_n^{k,*}$ might change up to $p$ times during each execution of the algorithm.

In the sequel, we will only consider actual weights $P_n^{k,*}$ that do not change by more than a positive constant $C_2$ at each update during the execution of a $p$-th maximal weight algorithm. This assumption can be justified in the same way as the assumptions of corollary 1 in section III *B*. As an actual weight $P_n^k$ is initialized as $P_n^{k,*} = P_n^k$ and as it is not updated more than $p$ times during the execution of the algorithm, there always holds $\forall n \geq 1$, $\forall k$, $1 \leq k \leq K$.

$$|P_n^k - P_n^{k,*}| \leq pC_2. \tag{16}$$

Similarly to theorem 1, we now prove a lower bound for the weight of the matching calculated a $p$-maximal weight matching algorithm:

**Theorem 3:** *For a network of IQ/CIOQ switches where each switch deploys p connections between each input and output pair and that applies a p-maximal weight matching algorithm, $p \geq 2$ with speedup $S = 1$, with positive weights $P^k > 0$, $1 \leq k \leq K$ that satisfy the condition (16), there holds for any timeslot n :*

$$\frac{p}{H}\sum_{k=1}^{K} P_n^k w^k \leq \sum_{k=1}^{K} P_n^k d_{n,s}^k + p^2 BNC_2\left(1 + \frac{1}{H}\right).$$

*Proof:* In its first iteration, the algorithm chooses the queue with the largest actual weight, say $P_n^{k_1,*}$ with $k_1 \in \mathcal{S}_{b_1,i_1,j_1}$, for transfer. Thus, from (11):

$$\begin{aligned} P_n^{k_1,*}H &\geq P_n^{k_1,*}\sum_{m\in\mathcal{S}_{b_1,i_1,j_1}} w^m \\ &\geq \sum_{m\in\mathcal{S}_{b_1,i_1,j_1}} P_n^{m,*}w^m. \end{aligned} \tag{17}$$

The weights $c_{i_1}$ and $c_{j_1}$ are updated. The second to the $p-1$-th iteration of the algorithm are performed exactly as the first iteration and allow us to derive inequalities as in (17). In the $p$-th iteration, after updating the weights, all logical queues $q^k$, $k \in \mathcal{S}_{b,i,j}$ such that $c_i = p \bigvee c_j = p$ are removed. In the $p+1$-th iteration, the remaining queue with the largest weight, say $P_n^{k_{p+1}}$ with $k_{p+1} \in \mathcal{S}_{b_{p+1},i_{p+1},j_{p+1}}$ is chosen. Thus,

$$\begin{aligned} P_n^{k_{p+1},*}H &\geq P_n^{k_{p+1},*}\sum_{\substack{m\in\mathcal{S}_{b_{p+1},i_{p+1},j_{p+1}}\\ m\notin\mathcal{S}_{b,i,j}\ s.t.\\ c_i=p\bigvee c_j=p}} w^m \\ &\geq \sum_{\substack{m\in\mathcal{S}_{b_{p+1},i_{p+1},j_{p+1}}\\ m\notin\mathcal{S}_{b,i,j}\ s.t.\\ c_i=p\bigvee c_j=p}} P_n^{m,*}w^m. \end{aligned} \tag{18}$$

The matching algorithm stops after $h$, $h \leq pBN$ iterations when either none or only empty queues remain. For each of the last $h-p$ iterations, an inequality analogous to (18) holds. For the eventually remaining empty queues, we hypothetically continue to run the algorithm $(pBN-h)$ times and produce valid inequalities as in (18) where both sides are equal to zero. Summing over all $BN$ inequalities of the types (17) and (18), we obtain:

$$\sum_{m=1}^{pBN} P_n^{k_m,*} \geq \frac{1}{H} \sum_{m=1}^{pBN} P_n^{k_m,*} w^k. \qquad (19)$$

Arguing as in the proof of corollary 1, we obtain from (7), (16), and (19):

$$\sum_{k=1}^{K} P_n^k d_n^k + p^2 BNC_2 \left(1+\frac{1}{H}\right) \quad \geq \quad \frac{p}{H} \sum_{k=1}^{K} P_n^k w^k. \Box$$

Using theorem 3, we can show that a network of IQ/CIOQ switches that implements a $p$-maximal weight matching scheduling policy achieves 100% throughput:

**Theorem 4:** *A network of IQ/CIOQ switches that implements a p-maximal weight matching scheduling policy, $p \geq 2$ with a speedup $S=1$, in which the weight $\phi_n^k$ of queue $q^k$ at time n is defined as in (8) and $\phi_n^k > 0$, achieves 100% throughput.*

*Proof:* The proof is given in the appendix.

## D. MNCM algorithms

MNCM algorithms have been introduced in [16]. They are based on maximum size matching algorithms that function on weight functions of the input and output ports rather than on weights functions of the $VOQ$s or the logical queues as the (p)- maximal weight matching algorithms described in the two previous sections. An MNCM algorithm is defined as follows:

**Definition 5:** A maximal size matching algorithm belongs to the class of MNCM algorithms if and only if each computed matching contains all nodes with maximum weight.

The existence of MNCM algorithms has been shown in [16]. An MNCM algorithm is executed without speedup and allows only one simultaneous connection between an input and and ouput, i.e.,

$$||D_n||_{IO} \leq 1. \qquad (20)$$

In order to define a specific set of MNCM algorithms, we introduce additional notation. By our assumption, the network contains $BN$ input and $BN$ output ports. We index all $2BN$ ports in the network with the index $h$, i.e. $1 \leq h \leq 2BN$. In analogy to the definition of per link weights in (8) and (9), we define per port weights $U_n^h$. For a port $h$ at switch $b$, we introduce the following notation:

$$\sum_{k \to h} = \begin{cases} \sum_{j=1}^{N} \sum_{k \in \mathcal{S}_{b,h,j},} & \text{if } h \text{ is an input port,} \\ \sum_{i=1}^{N} \sum_{k \in \mathcal{S}_{b,i,h}}, & \text{if } h \text{ is an output port.} \end{cases} \qquad (21)$$

The sum $\sum_{k\to h}$ runs over all logical queues at the switch $b$ that belong to input $h$/are destined to output $h$ . We set

$$\begin{aligned} \overline{F}_n^h &= \sum_{k \to h} \overline{d}_n^k, \qquad (22) \\ I^h &= \sum_{k \to h} w^k. \end{aligned}$$

We define a family of real positive functions $g_h(x) : \mathbb{N} \to \mathbb{R}$, $1 \leq h \leq 2BN$, that satisfy the following property:

$$\lim_{n\to\infty} \frac{g_h(n)}{n} = \frac{1}{I^h} \qquad \text{w.p.1.} \qquad (23)$$

For a given positive constant $G$, we define the weight of the port $h$ at time $n$ as:

$$U_n^h \quad = \quad n - g(\overline{F}^{(h)}(n)) + G. \qquad (24)$$

We set $U_n = (U_n^1, .., U_n^{2BN})$. Similar to the discussion of the constant $C$ (see (8)) in section III $A$, one can prove that for every function $g(\cdot)$ that satisfies the relation (23), one can always find a constant $G$, such that $U_n^h > 0$, $\forall h$, $1 \leq h \leq H$, $\forall n$, $n \geq 0$. Now, we state the following theorem:

**Theorem 5:** *A network of IQ/CIOQ switches that implements an MNCM scheduling policy with a speedup of $S=1$, in which the weights $U_n^h$ are chosen as in (24) and $U_n^h > 0$, achieves 100% throughput.*

*Proof:* The proof is given in the appendix.

## E. Networks of IQ/CIOQ switches that deploy different scheduling policies

In the three previous sections, three different scheduling policies for networks were presented. It was shown that they provide stability for a network of switches where all switches implement the respective scheduling policy.

In [2], the following stability result for a scheduling policy that is based on a maximum matching algorithm has been shown:

**Theorem 6:** *A network of IQ/CIOQ switches that implements a maximum weight matching scheduling policy with a speedup $S=1$, in which the weight $\phi_n^k$ of queue $q^k$ at time n is defined as in (9), achieves 100% throughput.*

In [2], theorem 6 is only proven for $C=0$. The proof can be easily generalized to the case $C \geq 0$. This policy is subject to constraint (20).

Following the argument in the previous section, it can be shown that a maximum weight matching policy can also be implemented in a distributed manner. It can now be asked if a network of switches in which each switch deploys any of the four scheduling policies stated in theorems 2, 4, 5 and 6 achieves 100% throughput. The answer is affirmative:

**Theorem 7** *A network of IQ/CIOQ switches where each switch deploys any of the policies defined in theorems 2, 4, 5 and 6 achieves 100% throughput.*

*Proof:* The proof is given in the appendix.

### *F. Distributed implementation of the scheduling algorithms*

In the sections $B$ - $E$, we presented three centralized scheduling algorithms for a network of switches. Now, we explain how those three scheduling algorithms can be implemented in a distributed manner. In a centralized implementation, a switch algorithm cannot implement any arbitrary matching. Depending on the scheduling algorithm, in each timeslot only a limited number of packets can be sent to each input and from each output. These limitations are expressed by the relations (10), (15) and (20) respectively. These relations prohibit specific configurations at individual switches. However, scheduling decisions taken at different switches do not constrain each other.

Further, for a maximal weight matching algorithm we see by its definition in section $B$, that when - in step 2 - a queue $q^{k_0}$ at switch $b$ is chosen, then - in step 3 - only queues that also belong to switch $b$ are removed. Thus, a maximal weight matching algorithms can be implemented in a distributed manner in the following way: The set of logical queues $q^k$ is split in subsets $S_b$ where $S_b$ contains the queues that belong to the physical queues of the switch $b$. Each switch $b$ executes the maximal weight matching algorithm on the set of logical queues that belong to $S_b$. The same argument holds for a $p$-maximal weight matching algorithm.

For an MNCM algorithm, we again divide the set of all logical queues in subsets $S_b$ and each switch $b$ executes the MNCM algorithm on the set of logical queues that belong to $S_b$. Whereas a centralized implementation guarantees that a matching contains all ports with maximum weight throughout the networks, a distributed implementation guarantees that at each switch all ports with maximum weights are contained in a matching. Obviously, a distributed implementation will satisfy the requirements of a centralized implementation as it selects all ports throughout the network with maximum weight. Thus, the proof of theorem 5 in the appendix is also valid for a distributed implementation.

## IV. Conclusions

This paper presents distributed scheduling policies of low complexity for networks of input-queued switches. In particular, it defines switching policies based on maximal weight matching, $p$-maximal weight matching and MNCM algorithms. It is shown that for specifc weights, a network of switches that deploys any of these three switching policies guarantees 100% throughput under admissible traffic. It is also proved that a network of input-queued switches where each switch implements either any of these three switching policies or a maximum weight matching based switching policy guarantees 100% throughput.

**Acknowledgement** The author thanks Dr.Vishal Sharma, Metanoia, Inc., as well as the referees for their comments and suggestions that helped to improve this paper.

## References

[1] Ajmone, M.M.,Leonardi, E., Mellia, M., Neri, F., *On the throughput achievable by isolated and interconnected input-queued switches under multicalss traffic,* Proc. of Infocom 2002, New York City, June 2002.

[2] Ajmone, M.M.,Giaccone, P., Leonardi, E., 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] Bauer, C., *Packet scheduling in input-queued switches with a speedup of less than two,* Proc. of IEEE International Conference on Networks, Sydney, Sept. 2003.

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

[6] 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.

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

[8] 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.

[9] 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.

[10] Leonardi, E., Mellia, M., Neri, F., Marsan, M.A., *On the stability of input-buffered cell switches with speed-up,* Proc. of IEEE Infocom 2000, Tel Aviv, Israel.

[11] 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.

[12] McKeown, N., Mekkittikul, A., Anantharam, V., Walrand, J., *Achieving 100% throughput in an input queued switch,* IEEE Transactions on Communications, vol. 47, no. 8, Aug. 1999, 1260 - 1272.

[13] McKeown, N., Mekkittikul, A., *A practical scheduling algorithm to achieve 100% throughput in input queued switches,* Proc. of IEEE Infocom 1998, San Francisco, March 1998.

[14] 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.

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

[16] Tabatabaee, V., Tassiulas, L., *MNCM a new class of efficient scheduling algorithms for input-buffered switches with no speedup,* Proc. of Infocom 2003, San Francisco, April 2003.

[17] Tarjan, R.E.; *Data structures and network algorithms.* Society for Industrial and Applied Mathematics, Philadelphia, PA, 1983.

[18] Yang, M., Zheng, S.Q., *An efficient scheduling algorithm for CIOQ switches with space-division multiplexing expansion,* Proc. of Infocom 2003, San Francisco, April 2003.

## V. Appendix: Proof of theorems 2, 4, 5 and 7

### *A. The fluid methodology*

The proofs of theorems 2,4 and 5 make use of the fluid methodology as introduced in [7], [8]. We now 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:

$$X(t) = X(0) + \Lambda t - D(t)(I - R), \qquad (25)$$

$$D(t) = \sum_{\pi \in \Pi} \pi T_\pi(t), \qquad (26)$$

$$\sum_{\pi\in\Pi} T_\pi(t) \quad = \quad t, \tag{27}$$

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) $\lim_{t\to\infty} f_k(t) \to t/w^k$, and that $\overline{d}^k(t) \to \infty$ for $t\to\infty$, we obtain

$$\phi^k(t) \to t - \frac{\overline{d}^k(t)}{w^k} + C. \tag{28}$$

*B. Proof of theorem 2*

We denote the set of all switch configurations found at time $t$ by maximal weight matching algorithms as defined in theorem 2 as $\prod_1(t)$. The fluid equations (6) and (7) can be rewritten as:

$$\dot{D}(t) \quad = \sum_{\pi_1\in\prod_1(t)} \pi_1 \dot{T}_{\pi_1}(t), \tag{29}$$

$$\sum_{\pi_1\in\prod_1(t)} T_{\pi_1}(t) \quad = \quad t. \tag{30}$$

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 (28), we obtain:

$$\dot{\Phi}(t) = \mathbb{I} - \dot{D}(t)\Gamma^{-1}. \tag{31}$$

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 $\forall t \geq 0$,

$$\Phi(t) \leq B, \tag{32}$$

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

$$\frac{d}{dt} V(\Phi(t)) \leq 0 \tag{33}$$

$\forall t$ 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 (32) for a certain $B > 0$. Thus we will show (33) in order to prove (32).

We note that for any $t$, there holds by the pigeonhole principle:

$$\max_{1\leq k\leq K} \phi^k(t) \geq \frac{\langle \mathbb{I}, \Phi(t)\rangle}{K}. \tag{34}$$

Finally, we express corollary 1 in the following way:

$$\frac{S}{H}\langle W, \Phi(t)\rangle \leq \langle \pi_1, \Phi(t)\rangle + K_1, \qquad \forall \pi_1 \in \Pi_1(t), \tag{35}$$

where $K_1 = \frac{S(S-1)}{2} K C_1 (1 + \frac{1}{H})$. Now (33) follows from (29), (30), (31), (34), and (35):

$$\begin{aligned}
&\frac{d}{dt}V(\Phi(t)) = \frac{1}{2}\langle \dot{\Phi}(t)\Gamma, \Phi(t)\rangle + \frac{1}{2}\langle \Phi(t)\Gamma, \dot{\Phi}(t)\rangle \\
= \quad & \langle \dot{\Phi}(t)\Gamma, \Phi(t)\rangle = \langle [\mathbb{I} - \dot{D}(t)\Gamma^{-1}]\Gamma, \Phi(t)\rangle \\
= \quad & \langle W, \Phi(t)\rangle - \sum_{\pi_1\in\prod_1(t)} \langle \pi_1 \dot{T}_{\pi_1}(t), \Phi(t)\rangle \\
= \quad & \langle W, \Phi(t)\rangle - \sum_{\pi_1\in\prod_1(t)} \dot{T}_{\pi_1}(t)\langle \pi_1, \Phi(t)\rangle \\
\leq \quad & \langle W, \Phi(t)\rangle \left(1 - \frac{S}{H}\right) + K_1 \\
\leq \quad & \left(1 - \frac{S}{H}\right) \min_{\substack{1\leq k\leq K\\ w^k>0}} w^k \frac{\langle \mathbb{I}, \Phi(t)\rangle}{K} + K_1 \\
\leq \quad & \frac{H-S}{2H} \min_{\substack{1\leq k\leq K\\ w^k>0}} w^k \frac{\langle \mathbb{I}, \Phi(t)\rangle}{K} \\
< \quad & 0,
\end{aligned}$$

where the second to last inequality holds for $\langle \mathbb{I}, \Phi(t)\rangle > K_0 > 0$ if $K_0$ is chosen sufficiently large. We see from (28) and (32):

$$0 < t - \frac{\overline{d}^k(t)}{w^k} + C \quad \leq \quad B. \tag{36}$$

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

$$\lim_{t\to\infty} \frac{D(t)}{t} \quad = \quad W, \qquad \text{w.p.1}, \tag{37}$$

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

*C. Proof of theorem 4*

We denote the set of all switch configurations found at time $t$ by maximal weight matching algorithms as defined in theorem 2 as $\prod_2(t)$. Theorem 3 can now be expressed as follows:

$$\frac{p}{H}\langle W, \Phi(t)\rangle \leq \langle \pi_2, \Phi(t)\rangle + K_2, \qquad \forall \pi_2 \in \Pi_2(t), \tag{38}$$

where $K_2 = p^2 BNC_2\left(1 + \frac{1}{H}\right)$. The proof of theorem 4 is identical to the proof of theorem 2 with the exception of using the relation (38) instead of the relation (35). The relations (29) and (30) hold with $\pi_1$ and $\prod_1(t)$ substituted by $\pi_2$ and $\prod_2(t)$. □

*D. Proof of theorem 5*

We first introduce some additional terminology. Set

$$F_n^h \quad = \quad \sum_{k\to h} d_n^k. \tag{39}$$

$\forall h$, $1 \leq h \leq 2NB$. Based on (22) and (39), we use the fluid methodology (see [7], [8]) to define the continuous function $F^h(t)$ and $\overline{F}^h(t)$ . By definition,

$$\frac{d(\overline{F}^h)_{t=t_0}}{dt} \quad = \quad F^h(t_0). \tag{40}$$

Arguing analogously to (28), we obtain

$$U^h(t) \to t - \frac{\overline{F}^h(t)}{I^h} + G.$$

Thus, we derive from (40)

$$\dot{U}^h(t) = 1 - \frac{\dot{\overline{F}}^h(t)}{I^h} = 1 - \frac{F^h(t)}{I^h}. \qquad (41)$$

Following an argument in [16], we introduce the function $f(U(t))$ that plays as similar role as the Lyapunov function (see (32)) in the proofs of theorems 2 and 4. The function $f(U(t)) : \mathbb{R}^{2BN} \to \mathbb{R}$ is defined as:

$$f(U(t)) = I^{a_t} \max_{1 \le h \le 2BN} (U^h(t))^2,$$

where $a_t = \arg \max_{1 \le h \le 2BN} U^h(t)$. As $U(t)$ is differentiable under the fluid model, $f(U(t))$ is continuous, but not necessarily differentiable. However, we can define the right derivative of $f(\cdot)$ as follows:

$$\frac{df(U(t))}{dt^+} = \lim_{\delta \to 0^+} \frac{f(U(t+\delta)) - f(U(t))}{\delta}.$$

We note that if $f(U(t)) = I^{h_0}(U^{h_0}(t))^2$, there holds:

$$F^{h_0}(t) = 1, \qquad (42)$$

This relation states that whenever a port $h$ has the maximum weight, a cell is scheduled for transfer from/to the input/output $h$. This follows from the definition of an MNCM algorithm.

We discuss the properties of the function $f(U(t))$ as in [16]. For every $t_0 > 0$, there is always a set of indices $Q_{t_0}$ such that $\forall h_0 \in Q_{t_0}$, $f(U(t_0)) = I^{h_0}(U^{h_0}(t))^2$. Moreover, for every $t_0 > 0$ there exists a $\delta > 0$ such that $f(U(t)) = I^{h_0}(U^{h_0}(t))^2$, $\forall t \in [t_0,\, t_0 + \delta]$ and $\forall h_0 \in Q_{t_0}$. We say that $U(t)$ is represented by $U^{h_0}(t)$, $h_0 \in Q_{t_0}$, in the interval $[t_0,\, t_0+\delta.]$ The time axis can always be partitioned into disjoint intervals $[0, \Delta_1[, [\Delta_1, \Delta_1 + \Delta_2[, ..$ such that each interval is represented by a common index $q_{i,\, i=1,2,...}$. If $f(U(t))$ is represented in the $i$-th interval by $q_i$, then

$$\left(\frac{df(U(t))}{dt^+}\right)_{t \in i-th\ int.} = 2\dot{U}^{h_{q_i}}(t) I^{h_{q_i}} U^{h_{q_i}}(t). \qquad (43)$$

Analogously to (32), we want to show that

$$U^h(t) \le B_1 \qquad (44)$$

for a certain $B_1 > 0$ and $\forall t$, $t \ge 0$, $\forall h$, $1 \le h \le H$. In order to prove (44), we show in analogy to (33) that

$$\left(\frac{df(U(t))}{dt^+}\right)_{t=t_0} \le 0,\ \forall t \ge 0. \qquad (45)$$

The relation (45) implies $f(U(t)) \le f(U(0))$, $\forall t$, $t \ge 0$, which in turn implies (44) with $B_1 = \sqrt{f(U(0))} / \min_{\substack{1 \le h \le 2BN \\ I^h > 0}} I^h$.

We assume that $t_0 \in i-$interval:

$$\begin{aligned}
\left(\frac{df(U(t))}{dt^+}\right)_{t=t_0} &= 2\dot{U}^{h_{q_i}}(t_0) I^{h_{q_i}} U^{h_{q_i}}(t_0) \\
&= 2\left(1 - \frac{F^{h_{q_i}}(t_0)}{I^{h_{q_i}}}\right) I^{h_{q_i}} U^{h_{q_i}}(t_0) \\
&= 2(I^{h_{q_i}} - 1) U^{h_{q_i}}(t_0) \\
&< 0.
\end{aligned}$$

The first and the second relation follow from (43) and (41) respectively. The third and fourth relations follow from (42) and (7) respectively. Arguing similarly as in (36) and (37), we derive from (44):

$$\lim_{t \to \infty} \frac{\overline{F}^h(t)}{t} = I^h, \qquad \forall h,\, 1 \le h \le H. \qquad (46)$$

From (46), the stability condition as given in (37) follows straight if we take into account that $\lim_{t \to \infty} \frac{\overline{d}^k(t)}{t} \le w^k$, $\forall k$, $1 \le k \le K$, which follows from the fact that the accumulative departure rate at each logical queue $q^k$ cannot be more than the accumulative arrival rate. □

## *E. Proof of theorem 7*

We divide the switches in the network into four groups $G_i$, $i \in \{1,2,3,4\}$ where the groups $G_1$, $G_2$, $G_3$ and $G_4$ contain the switches that deploy the switching policy defined in theorems 2, 4, 5, and 6, respectively. Accordingly, we can divide the departure vector $D(t)$ and the arrival rate vector $W$ in four subvectors, i.e., we write $D(t) = (D_1(t), D_3(t), D_3(t), D_4(t))$ and $W = (W_1, W_3, W_3, W_4)$. In order to prove the stability condition (37), it is sufficient to show that $\forall i \in \{1,2,3,4\}$

$$\lim_{t \to \infty} \frac{D_i(t)}{t} = W_i, \qquad \text{w.p.1.} \qquad (47)$$

For each $i \in \{1,2,3,4\}$, the relation (47) can be proved by applying the proofs of theorems 2, 4, 5 and 6 to the corresponding group of switches $G_i$ instead of applying them to the whole network of switches. □