Low Complexity, Stable Scheduling Algorithms for Networks of Input Queued Switches with No or Very Low Speed-up

Claus Bauer
Dolby Laboratories
100 Potrero Avenue
San Francisco, 94103, USA
cb@dolby.com

ABSTRACT

The delay and throughput characteristics of a packet switch depend mainly on the queueing scheme and the scheduling algorithm deployed at the switch. Early research on scheduling algorithms has mainly focused on maximum weight matching scheduling algorithms. It is known that maximum weight matching algorithms guarantee the stability of input-queued switches, but are impractical due to their high computational complexity. Later research showed that the less complex maximal matching algorithms can stabilize input-queued switches when they are deployed with a speed-up of two. For practical purposes, neither a high computational complexity nor a speed-up of two is desirable.

In this paper, we investigate the application of matching algorithms that approximate maximum weight matching algorithms to scheduling problems. We show that while having a low computational complexity, they guarantee the stability of input queued switches when they are deployed with a moderate speed-up.

In particular, we show that the $improve\_matching$ algorithm stabilizes input-queued switches when it is deployed with a speed-up of $\frac{3}{2} + \epsilon$.

In a second step, we further improve on these results by proposing a class of maximal weight matching algorithms that stabilize an input-queued switch without any speed-up.

Whereas initial research has only focused on scheduling algorithms that guarantee the stability of a single switch, recent work has shown how scheduling algorithms for single switches can be modified in order to design *distributed* scheduling algorithms that stabilize networks of input-queued switches. Using those results, we show that the switching algorithms proposed in this paper do not only stabilize a single switch, but also networks of input-queued switches.

Categories and Subject Descriptors

C.2 [**Computer-Communications Networks**]: General —*Data communications*; C.2.1 [**Network Architecture and Design**]: Packet-switching networks

General Terms

Modeling of communication networks, performance analysis

Keywords

Scheduling algorithms, stability

1. INTRODUCTION

The development of fast transmission technologies such as Dense Wavelength Division Multiplexing (DWDM) has led to an explosive increase in the transmission capacity of network links. In order to cope with the increased transmission capacities, network switches and routers must be equipped with high-speed switching and forwarding technologies. For this purpose, the choice of the scheduling algorithm is a major design criteria of a switch as it determines the throughput and delay characteristics of the switch. The scheduling algorithm defines a policy that determines which packets are forwarded from the inputs of the switch to the outputs of the switch and which packets are temporarily buffered at the input side.

As purely output-buffered switches have shown to be impractical due to the required high speed-up in the switching core, in this paper we only consider input-queued (IQ) or combined input/output (CIOQ) queued switches. A typical architecture of a IQ/CIOQ switch is given in figure 1. For each input i, there are N virtual output queues $VOQ_{i,j}$, $1 \leq j \leq N$. The cells arriving at input i and destined for output j are buffered in $VOQ_{i,j}$. The switching core itself is modeled as a crossbar which requires that not more than one packet can be sent simultaneously from the same input or to the same output. Many switch architectures require the switch to work at a speed-up of $S, S \geq 1$, which is defined as the ratio between the potentially higher speed of the switching core and the line-speed of the incoming (and outgoing) links.

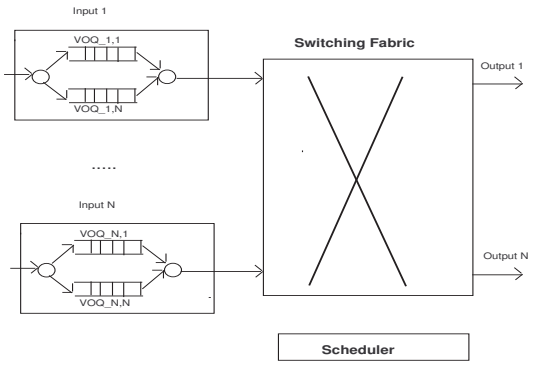

Figure 1: Architecture of an IQ/CIOQ queued switch

As a first milestone in research on scheduling algorithms, McKe-

own et al. [15] have shown that maximum weight matching (MWM) algorithms stabilize input IQ/CIOQ switches when they are deployed without speed-up, i.e., $S = 1$. However, their high computational complexity of $O(N^3)$ does not allow their practical deployment. As a less complex alternative, maximal matching (MM) algorithms have been investigated and it has been shown that they can stabilize IQ/CIOQ switches when they are deployed with a speed-up of $S = 2$. switches. The computational complexity of a maximal weight matching algorithm is $O(N^2)$.

Due to the impracticality of even low speedups at high line speeds, it is desirable to understand if scheduling algorithms exist that are of low complexity and can stabilize switches at speed-ups significantly smaller than $S = 2$ or even without a speed-up. The scope of this paper is to answer this question affirmatively and show both, the existence of stable algorithms at speed-ups smaller than $S = 2$ and in a second step the existence of MM algorithms that stabilize IQ/CIOQ switches without a speed-up, i.e., $S = 1$.

In order to establish these results, we proceed as follows. In the first step, we define a large class of MWM algorithms and show that they provide stability without speed-up. This investigation extends the work in [14] and is of independent interest, but also serves us procedurally to prepare our investigation of low complexity algorithms with low or no speed-up.

In a second step, we investigate the application of matching algorithms of low complexity that approximate MWM algorithms as scheduling algorithms for IQ/CIOQ switches. Generalizing the notion of a performance ratio from [11], we say that a matching algorithm approximates a MWM algorithm with *approximation parameters* (c, d), $0 < c \leq 1$, $d \geq 0$, if for any values of the weights of the matching algorithm, the sum of the weight calculated by the matching algorithm and of the constant d is at least c times as large as the optimal weight calculated by the MWM. The case $d = 0$ reduces the definition of *approximation parameters* to the concept of a performance ratio c as defined in [11]. The case $d > 0$ describes the case when the weight of the considered matching algorithm has to be increased by d in order to achieve a performance ratio c.

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

As a first application of these general results, we investigate four known types of maximal matching algorithms that have a computational complexity of $O(N^2)$. We derive a previously known result that their deployment with a rational speed-up $S \geq 2$ guarantees the stability of a IQ/CIOQ switch. Compared to previous work [3], [8], [6] that relied on a detailed analysis of the specific MM algorithm under consideration, our analysis of approximation MWM algorithms provides a unified framework to show the stability of the four considered types of MM algorithms.

As a second application, we discuss the $improve\_matching$ algorithm from [11] that approximates a MWM algorithm with approximation parameters $(\frac{2}{3} - \epsilon, 0)$ and has a computational complexity of $O(N^2)$. Applying our general results on approximation algorithms, we show that this algorithm stabilizes a IQ/CIOQ switch when it is deployed with a rational speed-up of $S \geq \frac{3}{2} + \epsilon$.

In the third step, we propose a class of MM algorithms that stabilize input IQ/CIOQ switches with a speed-up $S = 1$. In [19], it has been shown that under the assumption that no two $VOQs$ ever have the same occupancy, such a maximal weight matching algorithm exists. In practice, this assumption can obviously not be made. In contrast, this paper proposes the first maximal weight matching algorithms that stabilize a IQ/CIOQ switch without any speed-up and without assumptions on the occupancy $VOQs$. We prove the stability of the proposed algorithm by further developing an idea from [19]: We first define, as described above, a large class of maximum weight matching algorithms that guarantee the stability of a IQ/CIOQ switch. Then we show that a specific maximum weight matching algorithm out of this class is equivalent to the maximal weight matching algorithm we propose. This equivalence and the stability of the stability of the MWM algorithm prove the stability of the MM algorithm.

The improvement of our work compared to the ideas proposed in [19] derives from the specific choice of our weights. We define weights of the maximal weight matching algorithm such that at any point in time, no two weights are ever equal. This fact is crucial for showing the equivalence of the maximal matching algorithm with a specific maximum weight matching algorithm without making any assumptions on the occupancy of the $VOQs$.

The early work on switch algorithms quoted above has investigated stable scheduling algorithms for single IQ/CIOQ switches. Later work [1],[4] has shown that scheduling algorithms that guarantee the stability of a single switch might lead to instabilities when deployed in networks of IQ/CIOQ switches. In [1] and [4], switching policies that require the exchange of information between the switches have been proposed. In [2], for the first time a *distributed*, complex maximum weight matching algorithm that does not require the exchange of information between switches in the network has been proposed. We apply the methods developed in [2] to the design of the scheduling algorithms proposed in this paper. Consequently, the scheduling algorithms proposed here do not only guarantee the stability of a single switch, but also stabilize networks of IQ/CIOQ switches.

Specifically, we not only show that the proposed algorithms stabilize networks of IQ/CIOQ switches when all switches deploy the same scheduling algorithm, but we also prove that networks of IQ/CIOQ switches where each switch deploys any of the scheduling policies proposed in this paper are stable.

In conclusion, to the best knowledge of the author, this is the first time that stability for a network of IQ/CIOQ switches could be shown for a non-MWM algorithm with a speed-up strictly below $S = 2$. In particular, the class of MM algorithms proposed in this paper satisfies *the most common performance requirements on a scheduling algorithm:* It guarantees the stability of a network of switches, it can be implemented in a distributed manner that does not require the exchange of information between the switches, it is of feasible computational complexity, and it does not require any speed-up.

In sec. 3 - 5, we first describe the proposed scheduling policies in a non-distributed, centralized way, i.e., we will assume that the configurations of all switches are computed by a centralized server, which sends each switch its specific configuration. In sec. 7, we will then show how the algorithms presented in this paper can also be implemented in a distributed fashion, such that in each time slot, each switch calculates its own configuration. The stability results proved in this paper hold for both a centralized and a distributed implementation.

In the next section, we introduce a mathematical model of a network of switches. In sec. 3, we propose scheduling algorithms based on generalized MWM algorithms and provide stability results for these algorithms. Stable scheduling algorithms based on approximation algorithms are investigated in sec. 4. In sec. 5, we present a class of maximal weight matching algorithms that guarantees the stability of a network of IQ/CIOQ switches when it is

deployed with a speed-up $S = 1$. The stability of networks that deploy a combination of different types of scheduling algorithms at different switches is discussed in sec. 6. In sec. 7 we explain how the algorithms proposed in this paper can be implemented in a distributed manner throughout a network of switches. We conclude in sec. 8.

We mention that parts of the work presented in Section 4 have been described in a more condensed form in [5].

2. TERMINOLOGY AND MODEL

2.1 Model of a network of queues

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

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

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

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

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

$$E_n = A_n + T_n, \tag{2}$$

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, $r_{i,j} \in \{0, 1\}$,

$$max\left(\sum_{1\leq i\leq K} r_{i,j}, \sum_{1\leq j\leq K} r_{i,j}\right) \leq 1.$$

We set $r_{i,j} = 1$, 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 Λ. 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 ΛR^m. Noting that $(I - R)^{-1} = I + R + R^2 + ...$, we find that the overall average workload at the logical queues q^k is given by $W = \Lambda(I - R)^{-1}$.

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

DEFINITION 1. *A system of queues is rate stable if*

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

Throughout this paper, whenever we talk about a *stable* system, we refer to the notion of *rate stability* as given in definition 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 time slot 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:*

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

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:

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

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

We say that packets that enter the network via the input of a given switch and leave the network via the output of a given switch belong to the same flow. Packets belonging to the same flow travel through the same sequence of physical queues and are mapped to the same logical queues at each physical queue, i.e., a flow can be mapped bi-univocally 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 sec. 3 - 7 we prove the main results of this paper for *per-flow* scheduling schemes. In [2], it has been shown how *per flow* scheduling

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

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:*

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

Throughout this paper we assume a deterministic routing pattern defined by the routing matrix R and we call a traffic and routing pattern that satisfies the necessary rate condition for stability as defined in (5) *admissible*. Using definition 3, we can now rewrite the condition (5) 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:*

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

Without further mentioning, in the rest of this paper, we will only consider traffic and routing patterns that satisfy the condition (6).

3. MAXIMUM WEIGHT MATCHING POLICIES

3.1 Weight function

The scheduling policies introduced in this paper are based on matching algorithms. Any matching algorithm 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:

$$\lim_{n \to \infty} \frac{f_k(n)}{n} = \frac{1}{w^k}. \quad (7)$$

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

$$\phi_n^k \quad = \quad n - f_k(\overline{d}^k(n)) + c, \quad (8)$$

where c is a given constant, and we set $\Phi_n = (\phi_n^1, .., \phi_n^K)$. This choice of the weights is motivated by the fact that it will allow us to derive the relation (23) in the appendix, which in turn is used for the proof of the rate stability in Theorems 1 and 2. The constant c is chosen as follows: We see from (7) that there exists a constant c such that

$$f_k(\bar{d}_n^k) \geq \frac{\bar{d}_n^k}{w^k} - c. \quad (9)$$

As the cumulative departure rate is less than the cumulative arrival rate, there holds $\lim_{n \to \infty} \bar{d}_n^k \leq w^k n$. Both estimates imply that for a c chosen as in (9) the weight ϕ_n^k is always positive. This fact is used in the proofs of Theorems 1 and 2 in the appendix. For further usage, we note that the relations (7) and (8) imply that for any given positive integer v, there is

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

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

In [2], an example for $f_k(n)$ is given. The cumulative function of external arrivals for the logical queue q^k is given by $\overline{a}^k(n) = \sum_{m \leq n} a_m^k$. The inverse function $[\overline{a}^k]^{-1}(p)$ maps the packet number p to the time slot in which the packet arrived externally to the network. Setting $f_k(p) = [\overline{a}^k]^{-1}(p)$, the weight $\phi_n^k = n - [\overline{a}^k]^{-1}(p) + c$ denotes the age of the p-th packet at time n,i.e., the amount of time the packet has spent in the network, plus the constant c. At its departure time from the network n, the age of the p-th packet is $n - [\overline{a}^k]^{-1}(\overline{d}_n^k)$.

3.2 Definition of maximum weight matching algorithms

In this section, we define a class of maximum weight matching policies that guarantee the stability of a network of IQ/CIOQ switches. This class of policies is an extension of the policies defined in [14]. This extension is not only of interest in itself, but is necessary to prove the stability of a class of MM algorithms deployed without speed-up as specified in Theorem 5 below.
We define a set of functions $\mathcal{G}$:

DEFINITION 5. *A real function $F : \mathbb{R} \to \mathbb{R}$ is said to belong to the set $\mathcal{G}$ if*
a) F is monotonically non-decreasing, $F(0) = 0$, $F(x) > 0$ if $x > 0$.
b) $\dot{F}(x)$ exists for all $x > 0$.
c) $F(x) \to \infty$ for $x \to \infty$.

For a fixed set of functions $F_1, .., F_K \in \mathcal{G}$, we define the functional *weights*

$$\dot{F}_k(\phi_n^k),\ 1 \leq k \leq K. \quad (11)$$

Furthermore, we define the following vector function via the functions $F_1, .., F_K$ and their derivatives:

$$V_{\dot{F}}(\Phi(n)) \quad := \quad \left(\dot{F}_1(\phi_n^1), ..., \dot{F}_K(\phi_n^K) \right).$$

We write the scalar product for two vectors v_1 and v_2 as $\langle v_1, v_2 \rangle = v_1 v_2^T$. We define a schedule $\pi = (\pi^1, ..., \pi^K)$ of a switch as the chosen configuration of the switch core. If $\pi^k = 1$ then the logical queue q^k is connected to its output. In contrast if $\pi^k = 0$, then the logical queue q^k is not connected to its output.

We define a scheduling algorithm MWM^{V_F} as follows: At each time t, the scheduling algorithm MWM^{V_F} chooses the schedule π^{V_F} which is defined by the following equation:

$$\pi^{V_F}(n) \quad = \quad \arg \max_{\pi} \langle \pi, V_{\dot{F}}(\phi(n)) \rangle, \quad (12)$$

where the maximization is taken over all possible schedules π. This maximization problem is solved using the Hungarian method and

has a complexity of $O(K^3)$. We write the weight of the matching chosen by a MWM^{V_F} algorithm as defined in (12) as

$$D_n(MWM^{V_F})(V_{\dot{F}}(\phi(t)))^T = \max_{D_n} D_n(V_{\dot{F}}(\phi(t)))^T \quad (13)$$

We formulate the main result of this section:

THEOREM 1. *For any set of functions $F_1, .., F_K \in \mathcal{G}$, a network of IQ/CIOQ switches that implements a MWM^{V_F} scheduling policy is stable under admissible traffic.*

Proof: The proof is given in the appendix.

For further usage, we state the following corollary:

Corollary 1: If in (8) the weights are defined as $[\phi_k^n + 1]$ instead of ϕ_k^n, where $[x]$ denotes the biggest integer smaller than or equal to x, then Theorem 1 still holds.

Proof: The proof is nearly identical to the proof of Theorem 1.

4. APPROXIMATIONS TO THE MWM - ALGORITHM

In this section, we introduce local scheduling policies that are based on algorithms that approximate MWM algorithms.

4.1 Definition of an approximation MWM algorithm

We formally define an approximation MWM algorithm as follows.

DEFINITION 6. *For a scheduling algorithm $ALGO$ that approximates a MWM^{V_F} algorithm - as defined in (12) - with approximation parameters $\left(\frac{a}{b}, d\right), a, b \in \mathbb{N}$, a and b prime to each other, there holds*

$$\begin{aligned} & D_n(ALGO)(V_{\dot{F}}(\phi(t)))^T \\ \geq\ & \frac{a}{b} D_n(MWM^{V_F})(V_{\dot{F}}(\phi(t)))^T - d, \end{aligned} \quad (14)$$

where $D_n(ALGO)$ denotes the matching chosen by the algorithm $ALGO$.

4.2 The deployment of approximation algorithms in a switching core

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

We extend the notation introduced in sec. 2 as follows: We define $X_{n+\frac{da_1}{b_1}}$, $0 \leq d \leq b_1 - 1$, as the vector the entries of which are the number of packets buffered in the logical queues at time $n + \frac{da_1}{b_1}$. For every n satisfying $n \equiv (0 \, mod \, a_1)$, time $n + \frac{da_1}{b_1}$ denotes the beginning of the $(d+1)$ -th internal time slot after the the n-th external time slot. $D_{n+\frac{da_1}{b_1}}$ expresses the number of packets departing in the $(d+1)$-th internal time slot of the n-th external time slot, and $E_{n+\frac{da_1}{b_1}}$ is defined analogously.

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

For the $mode\_keep$, the evolution of the queue lengths is described as follows:

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

where $D_\delta = max\left(\underline{0}, dD_n - X_n - \sum_{0 \leq c < \frac{da_1}{b_1}} E_{n+c}\right)$. Here, $\underline{0}$ is the vector with K elements with all entries equal to zero and the maximum is taken for each vector entry separately. If $d_n^k = 1$, the entry d_δ^k denotes the difference between the number of cells that have been forwarded in the interval $[n, n + \frac{da_1}{b_1})$, and the number of internal time slots d in the interval. If $d_n^k = 0$, then $d_\delta^k = 0$.

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

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

where G_δ is a vector with K elements where each entry is an integer between 0 and b_1. If at time n, there holds $x_n^k \geq b_1$, then there is $g_\delta^k = 0$. If at time n, there holds $x_n^k < b_1$, then depending on the VOQ length x_n^k at time n and the arrival patterns e_{n+c}^k, $0 \leq c \leq d-1$, in the interval $[n, n + \frac{da_1}{b_1})$, the switch might not always be able to forward a packet from $VOQ^{i,j}$ even if the scheduling algorithm prescribes so because $d^k_{n+\frac{da_1}{b_1}} = 1$. The value g_δ^k equals the number of instances where this happens for the $VOQ^{i,j}$ and thus takes values c in the range $0 \leq c \leq b_1$.

The $mode\_keep$ mode requires less computations than the $mode\_reconfig$ mode. However, the $mode\_reconfig$ reacts faster to the changing lengths of the VOQ. Applying the analysis from [16], one can show that the $mode\_keep$ mode leads to larger average package delays at the $VOQs$ than the $mode\_reconfig$ mode.

In order to state the main result of this paper, we define a the set of functions $\mathcal{G}^*$ as a subset of the set $\mathcal{G}$ defined in section 3.2:

DEFINITION 7. *A real function F: $\mathbb{R} \to \mathbb{R}$ belongs to the set $\mathcal{G}^*$ if*

a) $F \in \mathcal{G}$.

b) $\ddot{F}$ exists and for any fixed positive constant c,

$$\lim_{x \to \infty} \max_{t \in [x-c, x+c]} \frac{\ddot{F}(x)}{\dot{F}(x)} = 0. \quad (15)$$

Now, we state the main result of this paper:

THEOREM 2. *We consider a network of IQ/CIOQ switches that implements an approximation MWM^{V_F} algorithm with approximation parameters $(\frac{a}{b}, d)$, with functional weights $F_k(\phi_n^k)$ as defined in (11), and where the functions $F_k \in \mathcal{G}^*$. Assuming admissible traffic, the network is stable when the MWM^{V_F} algorithm is deployed in either $mode\_keep$ or $mode\_reconfig$ with a rational speed-up of $S = \frac{b_1}{a_1} \geq \frac{b}{a}$.*

Proof: The proof is given in the appendix.

4.3 Examples of approximation algorithms

4.3.1 Maximal matching algorithms

The most common approximation algorithms to a MWM algorithm are variations of maximal matching algorithms. A maximal matching is a matching that is not properly contained in any other matching of the graph.

The greedy maximal matching algorithm works similar to a general maximal matching as explained in [12]. It differs from a general maximal matching algorithm by not choosing an arbitrary edge at each step, but picking the heaviest edge currently available instead. It has approximation parameters $(\frac{1}{2}, 0)$. A specific implementation for switches that deploy a $per - VOQ$ queueing and scheduling discipline is proposed in [3]. We generalize this implementation to a $per - flow$ queueing discipline as considered in this paper as follows:

DEFINITION 8. *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 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, we now formally define a maximal weight matching algorithm 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.*

Preis [17] presented another linear time approximation algorithm for a MWM algorithm with approximation parameters $(\frac{1}{2}, 0)$. The main idea is to replace the heaviest edge needed by the greedy algorithm with a locally heaviest edge.

A different approach is used by Drake and Hougardy in [9]. The main idea of the proposed algorithm is to grow in a greedy way two matchings independently and to return the heavier of both as a result. Again, this algorithm has approximation parameters $(\frac{1}{2}, 0)$. In [10], the same authors propose local improvements to a given matching as a postprocessing step to enhance the performance of the approximation algorithm for the MWM problem in practice. The postprocessing does not improve the approximation parameters $(\frac{1}{2}, 0)$.

Using the fact that all algorithms discussed in this section approximate the MWM algorithm with approximation parameters $\left(\frac{1}{2}, 0\right)$, we deduce from Theorem 2:

THEOREM 3. *For admissible traffic, the maximal matching algorithms described in this section stabilize a network of IQ/CIOQ switches when they are deployed in either $mode\_keep$ or $mode\_reconfig$ with a rational speed-up $S \geq 2$ and the functional weights are chosen as in (11).*

Thus, among others, we provide a new way to prove that greedy maximal weight matching algorithms are stable with a rational speed-up of $S \geq 2$ as shown in [3]. The maximal matching algorithms described in this section have a complexity of at most $O(K^2 \log K)$.

Algorithm $improve\_matching$
$(G = (V, E), w : E \to \mathbb{R}^+, M)$

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

Figure 2: *The $improve\_matching$ algorithm*

4.3.2 The $improve\_matching$ algorithm

We give a short overview of the main structure of the algorithm and refer the reader for the missing details to [11]. We assume that the reader is familiar with standard graph theoretic terminology as used in [11].

The idea of the $improve\_matching$ algorithm is first to use standard techniques to expand a given matching to a maximal matching (if the given matching is not already maximal) and then to make local improvements via appropriate augmentations to the maximal matching. In particular, the $improve\_matching$ algorithm considers only local improvements that are obtained via *short augmentations*. A *short augmentation* is defined as an augmentation such that all the edges in the augmenting set are adjacent to a specific edge of the graph.

Furthermore, the algorithm does not consider all *short augmentations*, but only considers *β-augmentations* , which are defined as those short augmentations that lead to a local gain of a factor of at least β, where β is a fixed constant > 1. In a particular instant, there might be more than one possible β-augmentation. Intuitively, it is desirable to choose the β-augmentation that produces the biggest gain. However, for the purpose of the $improve\_matching$ algorithm, it is sufficient to choose a *good β-augmentation*. A β-augmentation is called *good* if it achieves at least $(\beta - 1)/(\beta - \frac{1}{2})$ fraction of the gain that the best local β-approximation can achieve.

We now formally define the $improve\_matching$ algorithm in figure 2. First, the input matching M is made maximal (if necessary) and then no further changes are made to M. Instead, M is copied to M' and all local augmentations are done with respect to M'. The algorithm visits each edge $e \in M$ only once, and if it finds any β-augmentation set at this edge in M', it performs a good β-augmentation centered at e in M'. In [11], it is shown that the the complexity of the $improve\_matching$ is linear in the number of edges E, i.e., $O(K^2)$.

In order to achieve approximation parameters $\left(\frac{2}{3} - \epsilon, 0\right)$ the $improve\_matching$ algorithm is applied iteratively. We first use a maximal matching algorithm (see sec. 4.3.1) to calculate a maximal matching M_0 with a weight $w(M_0) \geq \frac{1}{2} w(M_{opt})$, where M_{opt} denotes a maximum weight matching of the graph $G = (V, E)$. We then apply the $improve\_matching$ algorithm to the matching M_0 to obtain a matching M_1 and then iteratively apply the algorithm to the matching M_i to obtain a matching M_{i+1}. It is shown in [11] that at most $O\left(\frac{1}{\epsilon}\right)$ iterations are required to achieve approximation parameters $\left(\frac{2}{3} - \epsilon, 0\right)$. Thus, we deduce from Theorem 2:

THEOREM 4. *The iterated $improve\_matching$ algorithm defined with functional weights as in (11) stabilizes a network of IQ/CIOQ switches under any admissible traffic when it is deployed with a rational speed-up $S \geq \frac{3}{2} + \epsilon$ in both modes $mode\_keep$ and $mode\_reconfig$.*

5. MAXIMAL WEIGHT MATCHING ALGORITHMS WITHOUT SPEED-UP

5.1 A stable maximal weight matching algorithm

In this section we define a class of maximal weight matching algorithms MM^{V_G} that guarantee the stability of a network of switches when they are deployed without a speed-up. A maximal weight matching algorithm with general weights has been described in sec. 4.3.1. Thus, in this section we only have to define the specific weights of the class of algorithms MM^{V_G}.

We consider a set of functions $g_i, 1 \leq i \leq K$ that belong to the set $\mathcal{G}$ defined in sec. 3.2. We also require that for $i \neq j$ and for any pair of two not necessarily different integers a and $b \in \mathbb{N}$, there holds

$$g_i(a) \neq g_j(b). \tag{16}$$

We define the functional weight of the queue q^k as

$$G_k\left(\phi^k(n)\right) \quad := \quad exp\left(g\left(\left[\phi^k(n)+1\right]\right)\right), \tag{17}$$

where $exp(x)$ is the exponential function. We note that the functions G_k in (17) correspond to the functions $\dot{F}_k$ in (11). Thus, the functions $F_k(x) \in \mathcal{G}$ introduced in sec. 3.2 correspond to the functions $\int_0^x G_k(t)dt$.

We now give an example for a set of functions g_k. A square-free number is defined as a natural number that cannot be divided by the square of any other natural number. We denote by $s_1 < s_2 < ... < s_K$ as the first K square-free numbers in increasing order. For a fixed even natural number m, we then define the function $g_k(x)$ as $g_k(x) = s_k x^m$. The definition of the weights ensures that (16) always holds. This is true because for any two different logical queues k and l, i.e., $k \neq l$, there exist a prime number p such that the largest power of p that divides $s_k([\phi^k(n)+1])^m$ is odd, whereas the largest power of p that divides $s_l([\phi^l(n)+1])^l$ is either even or equal to zero. In a specific implementation, the numbers s_i could be chosen as the first K prime numbers.

We now state the main result of this section:

THEOREM 5. *A network of IQ/CIOQ switches that implements a MM^{V_G}. scheduling policy with a speed-up $S = 1$ and the functional weights as defined as in (17) stabilizes a network of IQ/CIOQ switches under admissible traffic*

Proof: The proof is given in the appendix.

6. NETWORKS OF IQ/CIOQ SWITCHES WITH DIFFERENT SCHEDULING POLICIES

In sec. 3 - 5, we considered networks of IQ/CIOQ switches where all switches deploy the same scheduling algorithm. As a further extension, we show that a network of switches, where each switch in the network deploys any of the switching policies described in sec. 3 - 5, is stable as well:

THEOREM 6. *A network of IQ/CIOQ switches where each switch deploys any of the scheduling algorithms defined in Theorems 1 - 5 is stable under admissible traffic.*

Proof: The proof is given in the appendix.

7. DISTRIBUTED IMPLEMENTATION OF THE ALGORITHM

The scheduling algorithms as defined in sec. 3, 4, and 5 formulate the scheduling problem as an optimization problem that takes into account all logical queues of the network. Thus, these formulations assume the existence of a centralized scheduling algorithm that always knows the state of the whole network. This seems to contradict the purpose of the paper to investigate distributed scheduling policies, in which each switch only considers the logical queues at its own VOQs. In this section, we describe how the centralized scheduling policies proposed in sec. 3- 5 can be implemented in a distributed manner.

We first consider MWM scheduling policies. The maximization in (12) is subject only to the crossbar constraint: In each time slot, at each switch at most one cell can be sent from each one input and at most one cell can be sent to each output. However, a switch configuration at a specific switch does not constrain the choice of the switch configuration at another switch. Thus, we split the weight vector Φ into B weight sub-vectors $\Phi = (\Phi_1, .., \Phi_B)$, where the sub-weight vector Φ_b contains the logical queues at the b-th switch. Accordingly, we split the vector V_F in B sub-vectors $V_F = (V_{F,1}, .., V_{F,B})$. Thus, the maximization in (12) can be written as:

$$\begin{aligned} \pi^{\mathcal{V}^{\mathcal{F}}}(n) &= \arg\max_{\pi} \langle \pi, V_{\dot{F}}(\phi(n)) \rangle \\ &= \sum_{b=0}^{B} \arg\max_{\pi_b} \langle \pi_b, V_{\dot{F},b}(\Phi_b(n)) \rangle, \end{aligned}$$

where π_b is the schedule chosen at the b-th switch. The maximization problem $\max_{\pi_b} \left\{ \langle \pi_b, V_{\dot{F},b}(\phi_b(n)) \rangle \right\}$ can be solved solely at the b-th switch.

With regard to MWM approximation algorithms the same argument applies. Again, the approximation algorithm is executed separately at each switch in the network as the crossbar constraint. Finally, the same argument shows how the class of maximal matching algorithms MM^{V_G} can be implemented in a distributed manner.

8. CONCLUSIONS

This paper examines *distributed* scheduling algorithms of low complexity that stabilize networks of IQ/CIOQ switches with low or no speed-up and that do not require any coordination between the switches in the network.

First, we consider a generalized class of MWM algorithms that stabilize networks of IQ/CIOQ switches. Then, we investigate the application of approximation MWM algorithms scheduling algorithms for networks of IQ/CIOQ switches. We show that MWM approximation algorithms guarantee the stability of networks of IQ/CIOQ switches under specific speed-up requirements. Applying these results, we show that the $improved\_matching$ algorithm guarantees the stability of a network of IQ/CIOQ switches when it is deployed with a rational speed-up $S \geq \frac{3}{2} + \epsilon$.

Second, we propose a maximal matching algorithm that satisfies the most common performance requirements on a scheduling algorithm: It guarantees the stability of a network of switches, it can

be implemented in distributed manner that does not require the exchange of information between the switches, it is of feasible computational complexity, and it does not require any speed-up.

Finally, we prove that networks of IQ/CIOQ switches where each switch deploys any of the scheduling algorithms presented in this paper are stable.

9. REFERENCES

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

[2] Ajmone, M.M., Giaccone, P., Leonardi, E., Mellia, M., Neri, F., *Local scheduling policies in networks of packet switches with input queues,* Proc. of Infocom 2003, San Francisco, April 2003.

[3] Ajmone M.M., Leonardi, E., Mellia, M., Neri, F., *On the stability of input-buffer cell switches with speed-up,* Proc. of Infocom 2000, Tel Aviv, March 2000.

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

[5] Bauer, C., *Approximations to maximum weight matching scheduling algorithms of low complexity,* AICT 2005, Lisbon, Portugal.

[6] Bauer, C., *Throughput and delay bounds for input buffered switches using maximal weight matching algorithms and a speed-up of less than two,* Proc. of ICOIN 2004, Pusan, Korea, Springer LNCS 3090.

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

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

[9] Drake, D.E., Hougardy, S., *A simple approximation algorithm for the weighted matching problem,* Information Processing letters 85 (2003), 211-213.

[10] Drake, D.E., Hougardy, S., *Linear time local improvements for weighted matchings in graphs,* WEA 2003, LNCS 2647, Seiten 107-119, 2003.

[11] Drake, D. E., Hougardy, S., *A linear time approximation algorithm for weighted matchings in graphs,* ACM Transactions on Algorithms, 1(1), pages 107-122.

[12] Gabow, H.N., Tarjan, R.E., *Faster scaling algorithms for general graph-matching problems,* J. ACM 38:4, 1991, 815-853.

[13] Gabow, H.N., *Data structures for weighted matching and nearest common ancestors with linking,* SODA 1990, 434 - 443.

[14] McKeown, N., Keslassy, I., *Analysis of scheduling algorithms that provide* 100% *throughput in input-queued switches,* Proceedings of the 39th Annual Allerton Conference on Communication, Control, and Computing. Monticello, Illinois, October 2001.

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

[16] Neely, M.M., Modiano, E., Rohrs, C.E., *Tradeoffs in Delay Guarantees and Computation Complexity for NxN Packet Switches,* Proc. of the Conf. on Information Sciences and Systems, Princeton: March 2002.

[17] Preis, R., *Linear time 1/2 approximation algorithm for maximum weighted matching in general graphs,* Symposium on Theoretical Aspects of Computer Science (STACS) 1999, Springer LNCS 1563, 259 - 269.

[18] 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.

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

10. APPENDIX

10.1 The fluid methodology

For the proofs of the Theorems 1, 2, and 5, we use the fluid methodology and its extension given in [2] and [7]. Applying the definitions introduced in sec. 2, we define the three following continuous vector functions:

- $X(t) = (X_1(t), ..., X_K(t))$ denotes the number of packets in the logical queues at time t.
- $D = (D_1(t), ..., D_K(t))$ denotes the number of packet departures from the logical queues until time t.
- $A = (A_1(t), ..., A_K(t))$ denotes the number of packets arrivals at the logical queues until time t.

We consider a specific scheduling algorithm $\mathcal{F}$ and we define $\Pi_{\mathcal{F}} = \{\pi^{\mathcal{F}}\}$ as the set of all possible network-wide matchings chosen by $\mathcal{F}$. For all $\pi^{\mathcal{F}} \in \Pi_{\mathcal{F}}$, we denote by $T_{\pi}^{\mathcal{F}}(t)$ the cumulative amount of time that the matching $\pi^{\mathcal{F}}$ has been applied until time t by the algorithm $\mathcal{F}$. Obviously, $T_{\pi}^{\mathcal{F}}(0) = 0 \; \forall \pi^{\mathcal{F}} \in \Pi_{\mathcal{F}}$. Using (3), we obtain the fluid equations of the network of IQ/CIOQ switches as follows:

$$X(t) = X(0) + \Lambda t - D(t)(I - R), \quad (18)$$

$$D(t) = \sum_{\pi^{\mathcal{F}} \in \Pi_{\mathcal{F}}} \pi^{\mathcal{F}} T_{\pi}^{\mathcal{F}}(t), \quad (19)$$

$$\sum_{\pi^{\mathcal{F}} \in \Pi_{\mathcal{F}}} T_{\pi}^{\mathcal{F}}(t) = t. \quad (20)$$

The first equation models the evolution of the logical queues, whereas the second equation counts the total number of departures from the $VOQs$. The third equation reflects the fact that in each time slot each input is connected to some output. Taking the derivatives, we derive from (19) and (20):

$$\dot{D}(t) = \sum_{\pi^{\mathcal{F}} \in \Pi_{\mathcal{F}}} \pi^{\mathcal{F}} \dot{T}_{\pi}^{\mathcal{F}}(t), \quad (21)$$

$$\sum_{\pi^{\mathcal{F}} \in \Pi_{\mathcal{F}}} \dot{T}_{\pi}^{\mathcal{F}}(t) = 1. \quad (22)$$

Applying the fluid methodology further, we define a continuous version of the weights Φ_n - defined in (8) - as $\Phi(t)$.

10.2 Proof of Theorem 1

First, we state an algebraic relation which we will use for the subsequent proof. We note that by (7) $\lim_{t \to \infty} f_k(t) \to t/w^k$. Thus, by (8) and $\overline{d}^k(t) \to \infty$ for $t \to \infty$, we obtain

$$\phi^k(t) \to t - \frac{\overline{d}^k(t)}{w^k} + c, \quad (23)$$

from which we obtain by taking the derivative on both sides:

$$\dot{\Phi}(t) = \mathbb{I} - \dot{D}(t)\Gamma^{-1}, \tag{24}$$

where we define $\Gamma = [\gamma^{(i,j)}]$ as the diagonal matrix with $\gamma^{(k,k)} = w^k$, and let Γ^{-1} be the inverse of Γ. Then, we set $H(x) = \Gamma V_F(x),\ x \in \mathbb{R}^K$ and define the Lyapunov function:

$$G(t) = \langle \mathbb{I}, H(\Phi(t)) \rangle,$$

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

$$||\Phi(t)||_{IO} \leq B, \tag{25}$$

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

$$\frac{d}{dt}G(t) \leq 0 \tag{26}$$

$\forall t$ such that $||\Phi(t)||_{IO} > C$, then there holds $G(t) \leq \max_{s,\, ||G(s)||_{IO} \leq C} G(s)$, which by c) in definition 5 implies (25) for a certain $B > 0$. Thus, we will show (26) in order to prove (25). Now (26) follows from (24), (21), and (22):

$$\begin{aligned}
& \frac{d}{dt}G(t) \\
= \;& \frac{d}{dt}\langle 1, H(\Phi(t)) \rangle \\
= \;& \langle \dot{\Phi}(t), \Gamma V_{\dot{F}}(\Phi(t)) \rangle \\
= \;& \langle \mathbb{I} - \dot{D}(t)\Gamma^{-1}, \Gamma V_{\dot{F}}(\Phi(t))) \rangle \\
= \;& \langle W, V_{\dot{F}}(\Phi(t)) \rangle - \langle \dot{D}(t), V_{\dot{F}}(\Phi(t)) \rangle \\
= \;& \langle W, V_{\dot{F}}(\Phi(t)) \rangle - \langle \sum_{\pi^{V_F} \in \Pi_{V_F}} \pi^{V_F} \dot{T}_{\pi}^{V_F}(t), V_{\dot{F}}(\Phi(t)) \rangle \\
= \;& \langle W, V_{\dot{F}}(\Phi(t)) \rangle - \sum_{\pi^{V_F} \in \Pi_{V_F}} \dot{T}_{\pi}^{V_F}(t) \langle \pi^{V_F}(t), V_{\dot{F}}(\Phi(t)) \rangle \\
= \;& \langle W, V_{\dot{F}}(\Phi(t)) \rangle - \langle \pi^{V_F}(t), V_{\dot{F}}(\Phi(t)) \rangle \\
\leq \;& 0.
\end{aligned}$$

The last inequality follows from (12) and by an argument in [15]. We see from (23) and (25):

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

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

$$\lim_{t\to\infty} \frac{D(t)}{t} \;\; = \;\; W, \qquad \text{w.p.1,} \tag{27}$$

which corresponds to the rate stability condition of $X(t)$ according to definition 1. □

10.3 Proof of Theorem 2

10.3.1 Lower bounds for the weights calculated by approximation algorithms

In this section, we define lower bounds for the weight calculated by an approximation algorithm deployed with a rational speed-up $\frac{b_1}{a_1} > \frac{b}{a}$ in either $mode\_keep$ or $mode\_reconfig$. We will need these lower bounds for the proof of Theorem 2 in sec. 10.3.2. For our investigations, we consider the weight of all matchings calculated in a_1 successive time slots $[n, n+a_1)$ by an approximation algorithm with approximation parameters $(\frac{a}{b}, d)$ in $mode\_keep$. We find

$$\begin{aligned}
& \sum_{d=0}^{b_1-1} D_n(ALGO) V_{\dot{F}}^T(\Phi_{n+\frac{da_1}{b_1}}) \\
= \;& b_1 D_n(ALGO) V_{\dot{F}}^T(\Phi_n) \\
+ \;& D_n(ALGO) \sum_{d=0}^{b_1-1} \left(V_{\dot{F}}^T(\Phi_{n+\frac{da_1}{b_1}}) - V_{\dot{F}}^T(\Phi_n) \right).
\end{aligned} \tag{28}$$

We see from (10) and (15),

$$\begin{aligned}
& \left| D_n(ALGO)(V_{\dot{F}}^T(\Phi_{n+\frac{da_1}{b_1}}) - V_{\dot{F}}^T(\Phi_n)) \right| \\
\leq \;& K \max_{1\leq k\leq K} \left| (\dot{F}_k(\phi^k_{n+\frac{da_1}{b_1}}) - \dot{F}_k(\phi^k_n)) \right| \\
\leq \;& K \max_{1\leq k\leq K} \max_{n\leq t\leq n+\frac{da_1}{b_1}} \left| \ddot{F}_k(\phi^k_n) \right| \frac{da_1}{b_1} \\
\leq \;& \epsilon \max_{1\leq k\leq K} \dot{F}_k(\phi^k(n)),
\end{aligned} \tag{29}$$

for any arbitrarily small ϵ and for sufficiently large $\phi^k(n)$. Inserting (29) in (28) we see using (14)

$$\begin{aligned}
& \sum_{d=0}^{b_1-1} D_n(ALGO) V_{\dot{F}}^T(\Phi_{n+\frac{da_1}{b_1}}) \\
\geq \;& b_1 D_n(ALGO) V_{\dot{F}}^T(\Phi_n) - \epsilon \max_{1\leq k\leq K} \dot{F}(\phi^k_n) \\
\geq \;& \frac{ab_1}{b} D_n(MWM) V_{\dot{F}}^T(\Phi_n) - \epsilon \max_{1\leq k\leq K} \dot{F}_k(\phi^k_n) - b_1 d \\
\geq \;& a_1 D_n(MWM) V_{\dot{F}}^T(\Phi_n) - \epsilon \max_{1\leq k\leq K} \dot{F}(\phi^k_n) - b_1 d.
\end{aligned} \tag{30}$$

In a similar way we now derive a lower bound for an approximation algorithm with approximation parameters $(\frac{a}{b}, d)$ that is deployed in $mode\_reconfig$ with a rational speed-up $\frac{b_1}{a_1} > \frac{b}{a}$. For this purpose, we note that the relation (29) holds for any scheduling algorithm $ALGO$, in particular it holds for MWM^{V_F}. Using (14), (13), and (29), we obtain:

$$\begin{aligned}
& \sum_{d=0}^{b_1-1} D_{n+\frac{da_1}{b_1}}(ALGO) V_{\dot{F}}^T(\Phi_{n+\frac{da_1}{b_1}}) \\
\geq \;& \frac{a}{b} \sum_{d=0}^{b_1-1} D_{n+\frac{da_1}{b_1}}(MWM) V_{\dot{F}}^T(\Phi_{n+\frac{da_1}{b_1}}) - b_1 d \\
\geq \;& \frac{ab_1}{b} D_n(MWM) V_{\dot{F}}^T(\Phi_{n+\frac{da_1}{b_1}}) - b_1 d \\
= \;& \frac{ab_1}{b} D_n(MWM) V_{\dot{F}}^T(\Phi_n) - b_1 d \\
+ \;& \frac{ab_1}{b} D_n(MWM) \left(V_{\dot{F}}^T(\Phi_{n+\frac{da_1}{b_1}}) - V_{\dot{F}}^T(\Phi_n) \right) \\
\geq \;& a_1 D_n(MWM) V_{\dot{F}}^T(\Phi_n^T) - b_1 d - \epsilon \max_{1\leq k\leq K} \dot{F}_k(\phi^k(n)).
\end{aligned} \tag{31}$$

10.3.2 Proof of stability

Applying the principles of the fluid terminology as in [7] and dividing both sides of the equations by the number of considered time slots a_1, we express the equations (28) and (31) in the fluid

terminology as follows:

$$\begin{aligned} & \langle \pi^{V_F}(t), \Phi(t) \rangle \\ \leq \; & \langle \pi^{ALGO}(t), \Phi(t) \rangle + K_1 + \epsilon \max_{1 \leq k \leq K} \dot{F}_k(\phi^k(t)), \quad (32) \end{aligned}$$

where $\pi^{V_F}(t)$ and $\pi^{ALGO}(t)$ are the matchings chosen at time t by MWM^{V_F} and $ALGO$ algorithms, respectively, $\epsilon > 0$ can be chosen arbitrarily small, and $K_1 = b_1 d$. Now the proof of Theorem 2 is similar to the proof of Theorem 1 by taking into account (32):

$$\begin{aligned} \frac{d}{dt}G(t) &= \frac{d}{dt}\langle 1, H(\Phi(t)) \rangle \\ &= \langle \dot{\Phi}(t), \Gamma V_{\dot{F}}(\Phi(t)) \rangle \\ &= \langle \mathbb{I} - \dot{D}(t)\Gamma^{-1}, \Gamma V_{\dot{F}}(\Phi(t))) \rangle \\ &= \langle W, V_{\dot{F}}(\Phi(t)) \rangle - \langle \dot{D}(t), V_{\dot{F}}(\Phi(t)) \rangle \\ &= \langle W, V_{\dot{F}}(\Phi(t)) \rangle \\ &- \langle \sum_{\pi^{ALGO} \in \Pi_{ALGO}} \pi^{ALGO} \dot{T}_\pi^{ALGO}(t), V_{\dot{F}}(\Phi(t)) \rangle \\ &\leq \langle W, V_{\dot{F}}(\Phi(t)) \rangle + K_1 + \epsilon \max_{1 \leq k \leq K} \dot{F}_k(\phi(n)) \\ &- \langle \sum_{\pi^{ALGO} \in \Pi_{ALGO}} \pi^{V_F}(t) \dot{T}_\pi^{ALGO}(t), V_{\dot{F}}(\Phi(t)) \rangle \\ &\leq \langle W, V_{\dot{F}}(\Phi(t)) \rangle + K_1 + \epsilon \max_{1 \leq k \leq K} \dot{F}_k(\phi(n)) \\ &- \sum_{\pi^{ALGO} \in \Pi_{ALGO}} \dot{T}_\pi^{ALGO}(t) \langle \pi^{V_F}, V_{\dot{F}}(\Phi(t)) \rangle \\ &= \langle W, V_{\dot{F}}(\Phi(t)) \rangle - \langle \pi^{V_F}, V_{\dot{F}}(\Phi(t)) \rangle \\ &+ K_1 + \epsilon \max_{1 \leq k \leq K} \dot{F}_k(\phi(n)). \quad (33) \end{aligned}$$

We see from (6) that there exists a constant $\epsilon_1 > 0$ such that $||W||_{IO} \leq 1 - \epsilon_1$. Applying (12), we obtain from (33)

$$\begin{aligned} & \frac{d}{dt}G(t) \\ = \; & \langle V_{\dot{F}}(\Phi(t)), W - (1-\epsilon_1)\pi^{V_F} \rangle - \epsilon_1 \langle \pi^{V_F}, V_{\dot{F}}(\Phi(t)) \rangle \\ + \; & K_1 + \epsilon \max_{1 \leq k \leq K} \dot{F}_k(\phi(n)) \\ \leq \; & -\epsilon_1 \langle \pi^{V_F}, V_{\dot{F}}(\Phi(t)) \rangle + K_1 + \epsilon \max_{1 \leq k \leq K} \dot{F}_k(\phi(n)). \quad (34) \end{aligned}$$

The last estimate is derived using a well-known argument based on Birkhoff's theorem as in [15]. We note that

$$\dot{F}_{k_{max}}(\phi(n)) := \max_{1 \leq k \leq K} \dot{F}_k(\phi(n)) \leq \langle \pi^{V_F}, V_{\dot{F}}(\Phi(t)) \rangle. \quad (35)$$

Otherwise there would be

$$\langle \pi^*, \dot{F}_{k_{max}}(\phi(n)) \rangle > \langle \pi^{V_F}, V_{\dot{F}}(\Phi(t)) \rangle, \quad (36)$$

where π^* is the schedule defined as $\pi^{k_{max}} = 1$ and $\pi^k = 0$ for $k \neq k_{max}$. This is impossible as the relation (36) would contradict the definition of the MWM algorithm MWM^{V_F}. Choosing $\epsilon = \epsilon_1/2$, we see from (34) and (35):

$$\frac{d}{dt}G(t) \leq -\frac{\epsilon_1}{3} \langle \pi^{V_F}, V_{\dot{F}}(\Phi(t)) \rangle < 0,$$

for a sufficiently large $||\Phi(t)||_{IO}$. Now Theorem 2 follows from (23) and (25) in the same way as Theorem 1.□

10.4 Proof of Theorem 5

We follow an idea in [19] to show that the algorithm MM^{V_G} always calculates the same matching as the following maximum weight matching algorithm MWM^{V_G} : MWM^{V_G} is defined as the MWM algorithm with the functional weights chosen as in (17). This maximum weight matching algorithm is stable by Corollary 1. The stability of MM^{V_G} then follows from the fact that MWM^{V_G} and MM^{V_G} always calculate the same matching. This is shown using an argument from [19]. In the first iteration, the MM^{V_G} algorithm chooses the queue with the largest functional weight of all $G_k(\phi^k(n))$, $1 \leq k \leq K$, say $G_a(\phi^a(n))$. We recall from (16) and (17) that at any time n, there is $g_a([\phi^a(n)+1]) \neq g_b\left(\left[\phi^b(n)+1\right]\right)$. Thus, the $MWM^{V_{g,E}}$ algorithm, which maximizes the weight of the whole matching, will also choose the queue q^a with the functional weight $G_a(\phi^a(n))$ for packet transfer because

$$\begin{aligned} \sum_{\substack{k=1 \\ k \neq a}}^{K} G_k(\phi^k(n)) &= \sum_{\substack{k=1 \\ k \neq a}}^{K} exp(g_k\left(\left[\phi^k(n)+1\right]\right) \\ &\leq \sum_{b=1}^{g([\phi^a(n)+1])-1} exp(b) \\ &= \frac{exp([\phi^a(n)+1]) - 1}{exp(1) - 1} \\ &< exp([\phi^a(n)+1]) \\ &= G_a(\phi^a(n)). \end{aligned}$$

Due to the crossbar structure of the switch, neither the MM^{V_G} nor the MWM^{V_G} algorithm chooses any logical queue for packet transfer that competes for a switch input and output with the chosen logical queue. Thus, these logical queues can be discarded for the rest of the proof. Applying the same arguments to the subset of the remaining queues, one sees that both algorithms choose the queue with the largest functional weight among the remaining queues. The successive application of this argument shows that both scheduling algorithms are indeed identical.

10.5 Proof of Theorem 6

In order to prove Theorem 6, we apply the techniques used for the proofs of Theorems 1- 5. We divide the switches in the network into h groups G_h where h denotes the number of different switching policies deployed throughout the considered network of IQ/CIOQ switches. Accordingly, we divide the departure vector $D(t)$ and the arrival rate vector W into h sub-vectors, i.e., $D(t) = (D_i(t))_{1 \leq i \leq h}$ and $W = (W_i)_{1 \leq i \leq h}$. In order to prove stability according to def. 1, we have to show that for $1 \leq i \leq h$,

$$\frac{D_i(t)}{t} = W_i, \;\; w.p.1. \quad (37)$$

For each i, the relation (37) can be shown by applying the respective proofs of the Theorems 1, 2, and 5 to the respective group of switches G_i, instead of applying them to the whole network of switches.