A Framework for Differential Frame-Based Matching Algorithms in Input-Queued Switches

Andrea Bianco, Paolo Giaccone, Emilio Leonardi, Fabio Neri
Dipartimento di Elettronica, Politecnico di Torino, Italy
e-mail: {bianco,giaccone,leonardi,neri}@mail.tlc.polito.it

Abstract— **We propose a novel framework to solve the problem of scheduling packets in high-speed input-queued switches with frame-based control. Our approach is based on the application of game theory concepts. We define a flexible scheduling policy, named SSB (Slot Sell and Buy): the existence of a unique Nash equilibrium for the policy is proved, together with properties of convergence of these equilibria. These findings allows us to state that our SSB scheduling policy achieves 100% throughput both in isolated input-queued switches and in networks of input-queued switches. Simulation results are used to further validate the approach and to show its flexibility in dealing with differentiated QoS guarantees.**

I. Introduction

In the last years a lot of attention has been devoted to scalability issues in switching architectures, the main reason being the continuously increasing speed of transmission lines, and the current trend towards convergence and consolidation of networking technologies. IQ (Input Queued) and CIOQ (Combined Input-Output Queued) switches have better intrinsic scalability properties with respect to output queued and shared memory architectures, since they do not require an internal increase in speed, named speedup, with respect to input line speed, in both the switching fabric and in the buffer used to store packets waiting to be forwarded.

However, in IQ and CIOQ switches two problems arise: first, a rather complex queue architecture (named VOQ, Virtual Output Queueing) is required to obtain good performance and, second, the access to the switching fabric must be controlled by some form of scheduling algorithm which operates on the state of input queues. The scheduling algorithm is often called matching algorithm due to the equivalence with matching in bipartite graphs. This means that control information must be exchanged among line cards, either through an additional data path or through the switching fabric itself, and that intelligence must be devoted to the scheduling algorithm, either at a centralized scheduler, or at line cards in a distributed manner. The scheduling algorithm complexity is usually the main performance limitation, since the time available to run the scheduling algorithm decreases linearly with the line speed (at 10Gbit/s a 64-bytes packet lasts about 50ns). Thus, to achieve good scalability in terms of switch size and port data rate, it is essential to reduce the computational complexity of the scheduling algorithm, as already pointed out by other researchers (e.g., see [1]).

As most researchers, we refer in this paper (i) to the case of switches operating on fixed-size cells, (ii) to a synchronous switch operation according to the cell time named slot, and (iii) to a bufferless switching fabric.

Several slot-by-slot matching algorithms have been proposed in the last years [2], [3], [4], [5], [6], [7], [8]. In all those schemes, the switching configuration is selected at each slot with independent decisions based upon the instantaneous state of input queues. This means that during one time slot the information about the queue state must be communicated from the input cards to the scheduler, which runs the algorithm to compute the matching. This approach does not scale well when slot times keep reducing due to the increasing data rates.

Two possible approaches can be pursued to control and reduce the matching complexity. First, cells transmissions can be organized in a frame, and matching algorithms can be run once in a time frame [1], [9], [10]. Second, given the correlation in time of queue occupancies, memory of the previous frame can be exploited in a differential scheme to reduce matching complexity [6], [8].

In a frame-based matching scheme, a frame length is defined in terms of a number of slots, and the scheduler acts on snapshots of queues taken at frame boundaries. A matching algorithm is run once in a time frame to obtain the switch configuration in each time slot inside a frame. Thus, the matching algorithm can be run on a time-scale largely independent from the line speed, since the frame length can be tailored by the designer on the basis of the running time of the algorithm, and of the required performance figures (delays tend to increase in frame-based schemes). In other words, this approach permits a conceptually important decoupling of the time scale upon which the switch is controlled from the time scale upon which individual data unit are transmitted, i.e., a decoupling of the time constraints of the control plane and the data plane.

Differential matching exploits the queues state correlation to ease the task of the matching algorithm. This approach was pursued in [6], [8], where the new matching is obtained by comparing simple perturbations of the matching selected in previous slot.

It has been recently shown that schedulers based on MWM fail to guarantee 100% throughput in *networks* of interconnected IQ switches. Hence, new policies suited for networks of interconnected switches were proposed and proved to achieve 100% throughput (see [11], [12], [13]). The significance of those results, however, is mainly theoretical since the proposed

This work was supported by the Italian Ministry for Education, University and Research, within the TANGO project.

policies appear too complex for a practical implementation in high-speed routers. Moreover, most of the policies defined in [11], [12], [13] require some form of active coordination and cooperation among different switches, thus either the definition of a new signalling protocol among switches, or the introduction of new fields in the packet format, are required to support them in real networks. As a consequence, the definition of an optimal scheduling policy that guarantees 100% throughput for networks of IQ switches and can be efficiently implemented is an interesting research problem.

In this paper we propose a novel approach based on a differential frame-based matching scheme, trying to exploit both the frame-based advantages, and the features of differential schemes. To achieve good performance, both in terms of throughput and delay, we design an adaptive scheduling scheme in which the switch configuration sequence is kept dynamically matched to the traffic traversing the switch. A simple closed-loop scheme that is in charge of adapting the scheduling sequence to traffic dynamics is proposed.

We were able to model our scheme in the game theory framework. If we see all the possible switch configurations as possible resource allocations, we will be able to model the frame-based matching problem as a resource allocation problem, where each virtual output queue plays the role of an agent who is trying to buy the switching resources. According to this model, input cards play a competitive game to decide their share of the switching fabric bandwidth. We apply the game theory framework to prove the properties of our scheduling algorithm. A unique Nash Equilibrium Point (NEP) [14] can be proved to exist for the game that models the scheduling algorithm evolution. We further prove the efficiency of the unique NEP, and the convergence of the algorithm to the NEP. Using these intermediate results, we are able to prove that our scheduling policy achieves 100% throughput under a large class of input traffic patterns, and that this result holds also in a network of switches. Moreover, we examine the proposed framework in realistic scenarios by simulation, proving the system flexibility and good performance.

The contribution of the paper, beyond the particular proposed frame-based scheduler, lies in the general framework in which such scheduler is presented. Although the stability properties and the performance of the proposed scheme can probably be obtained with simpler heuristics, our framework bears significant flexibility, and enables generalizations to other interesting performance metrics (i.e., to other cost functions), while preserving the provability of optimality.

II. Frame-Based Scheduling

Our system is based on a differential frame-based matching technique. Before providing an overview of the proposed algorithm, described in detail in Sect. IV, we better motivate our approach and describe more precisely the frame based matching environment.

In a frame-based matching scheme, a frame is defined as a sequence of F consecutive time slots, and the selection of a set of non conflicting cells is performed at the end of each frame, typically on the basis of snapshots of queues state taken at frame boundaries. Thus, the first advantage of frame-based algorithms is that they do not run at the time slot scale since they are executed at frame boundaries.

Note that, whereas the maximum number of non conflicting cells set is equal to N in an $N \times N$ switch for slot by slot algorithms, in the case of frame-based matching the maximum number is $N \times F$. Thus, in frame-based matching, a second step, often named scheduling, is required after the matching procedure, to determine the order in which the set of non conflicting data are transferred in the sequence of time slots within the frame. It is known that the asymptotic sequential complexity of slot-by-slot matching and frame-based matching are comparable, and varies depending on the considered algorithm (between $O(N^3)$ and $O(N^2)$); however, the complexity of frame-based matching may depend also on the frame size F.

For what regards complexity, the best compromise between complexity and performance has been an open and questionable issue in the design of schedulers. It is however important to note that heuristic matching schemes with acceptable complexities were shown to be more robust to traffic scenarios and provide better performance for frame-based schedulers with respect to slot-by-slot schedulers (see, e.g., [10]).

The frame length is an important system parameter that must be chosen with care, jointly with the slot size. Discussing this issue is beyond the scope of this paper, but typical slot sizes are around one hundred bytes, and the frame lengths are around hundreds/thousands of slots. Fixed-size frames are preferred to variable-size frames since they are best suited to be implemented (their running time is fixed) and to provide bandwidth guarantees to flows: indeed, bandwidth requests can be easily translated in slots per frame. We consider only fixed-size frames in this paper.

Frame-based matching may entail an increase in cell transfer delay through the switch with respect to slot-by-slot matching; the delay is tied to the frame length, and this effect is evident at low loads. However, given the higher efficiency of frame-based scheduling with respect to slot-by-slot scheduling (when the algorithms have similar asymptotic sequential complexity), it can be shown that they may provide higher throughput, and thus also lower delays at high loads [10]. Although the continuously increasing line speed makes the slot time shorter when measured in seconds, end-to-end delay guarantees must be satisfied on an absolute basis (e.g., milliseconds), making easier the task of satisfying delay guarantees within faster switches.

A final advantage of frame-based matching is the reduction in the amount of information that must be exchanged through the switch to run the algorithm. Even if the difference heavily depends on the considered algorithm, the need to exchange less information depends directly from the fact that frame-based matching algorithms need to know the system state less frequently than slot-by-slot algorithms. Differential approaches, where successive matchings are computed by small perturbations of previous matchings, help in further reducing the amount of information that must be exchanged among line cards to run matching algorithms.

For all the above reasons, we base our proposal on a differential fixed-size frame-based matching algorithm.

We assume that traffic rate estimates are obtained on-line (i.e. in real time) by using a proper measurement algorithm; the details of this estimate process are beyond the scope of this paper, but a standard numeric filter based on an exponential weighted average can be used. Note that the estimation may derived relying on a much simpler use of the queue length information only; this choice, which is very attractive for implementation, does not permit a theoretical analysis to be performed, but proves to perform very well in simulation.

We take a differential approach: rather than using the latest traffic estimates to determine a new matching, we use this information to modify the matchings determined in the previous frame, thereby reducing the algorithmic complexity due to traffic correlation among frames.

The algorithm used to modify the matchings is based on a negotiation process run among input cards, with the goal of "purchasing" the right amount of bandwidth. Each input/output pair minimizes a cost function; this process may increase, decrease or keep constant the amount of slots assigned to each input/output pair within the frame. Note that the algorithm uses traffic rate estimates only to adapt the previous frame to the modified traffic scenario; thus, it is more suited to a scenario where traffic rates do not vary dramatically within a frame time, which however is a reasonable assumption especially for core routers aggregating thousands of flows. This choice allows us to control algorithmic complexity at the price of a reduced system reactivity. However, the system proves to provide good performance also when loaded with time-variant traffic, as shown using simulation results.

III. System Model and Notation

We consider IQ or CIOQ cell-based switches with N input ports and N output ports, all running at the same cell rate. The switching fabric is assumed to be non-blocking and memoryless, i.e., cells are only stored at switch inputs and outputs. The average cell arrival rate at input i and directed to output j (with $i, j = 0, 1, 2, \dots, N-1$) is denoted by λ_{ij}; the average traffic matrix is $\Lambda = [\lambda_{ij}]$.

At each input, cells are stored according to the VOQ policy: one separate queue is maintained at each input for each output. Thus, the total number of input queues in each switch is N^2. We do not model possible output queues since in this paper we are mainly interested in studying the transfer of cells from inputs card to output cards through the switching fabric. Neither we consider multiple traffic classes. Let q_{ij} be the queue at input i storing cells directed to output j. L_{ij} denotes the number of cells stored in queue q_{ij}.

We consider a synchronous operation, in which the switch configuration can be changed only at slot boundaries. We call *internal time slot* the time necessary to transmit a cell from an input toward an output. We call instead *external time slot* the duration of a cell on input and output lines. The difference between external and internal time slots is due to the switch speed-up, and to possibly different cell formats (e.g., due to additional internal header fields).

We consider crossbar-like switching fabrics, i.e., at each internal time slot the switching fabric supports up to N parallel connections among inputs and outputs on which cells are simultaneously transferred. Let $S = [s_{ij}]$ be an $N \times N$ binary matrix which represents the switching configuration: $s_{ij} = 1$ if the switching fabric connects input i with output j; otherwise, $s_{ij} = 0$. We typically assume that S represent an input/output permutation, i.e. $\sum_k s_{ik} = 1$ and $\sum_k s_{kj} = 1$ for all i and j.

To avoid overloading any input and output port, the total average arrival rates in cells per external slot must be less than 1 for all input and output ports; in this case we say that the traffic loading the switch is *admissible*, that is $\max_k\{\sum_k \lambda_{ik}, \sum_k \lambda_{kj}\} < 1$.

Note that any admissible traffic pattern can be transferred without losses in an OQ switch with large enough queues.

IV. Scheduling Policy Description

The scheduling policy we propose is called SSB (Slot-Sell-and-Buy) and operates on a fixed-frame horizon; let F denote the frame length in slots. At the beginning of each frame, the scheduling algorithm is in charge of computing the scheduling plan, i.e., the set $\mathcal{S}$ of switching matrices $S(k)$, with $0 \le k < F$, for all the F slots within the frame.

The scheduling algorithm works in two phases:

1) **Bandwidth-Computation**. At the beginning of each frame, matrix $B = [b_{ij}]$ is evaluated; the element b_{ij} denotes the effective bandwidth allocated in the current frame to input/output pair (i, j), normalized to the external line rate.
2) **Plan-Scheduling**. Given B from the previous phase, the F switching matrices $S(k)$ of the scheduling plan $\mathcal{S}$ are computed for the current frame, in such a way that $\sum_{k=0}^{F-1} s_{ij}(k) = b_{ij} \times F$.

We now describe the two phases in more detail.

A. Bandwidth-Computation Phase

We consider a simple algorithm for the evaluation of bandwidths b_{ij} assigned to each input/output pair; this algorithm is conceptually inspired by game theory concepts.

In our game, a different agent corresponds to each input/output pair. A central scheduler interacts with the agents. The algorithm is run once per frame, e.g., at the beginning of the frame. Each agent eventually buys a given amount of bandwidth x_{ij}; we name $X = [x_{ij}]$ the negotiated bandwidth matrix. The central scheduler determines matrix B, which is dependent on X through a *bandwidth allocation* function.

The agent decision is taken on the basis of bandwidth needs (possibly determined via some type of measurement in the previous frame) and bandwidth costs (issued by the central scheduler). Agents are constrained to limit their bandwidth increase/decrease requests per frame to increment/decrement with respect to the bandwidth assigned to each agent in the previous frame; thus, the scheduler may run a differential algorithm in the plan-scheduling phase, limiting the scheduling complexity.

The central scheduler, on the basis of the knowledge of the negotiated bandwidth matrix X assigns a cost per slot per frame to each agent. The cost per slot is communicated to each agent, who will base its next negotiation evaluation on the basis of the new costs and of its needs. Note that the agent has no knowledge of the bandwidth purchased by other agents; the system is such that an increment (decrement) of the purchased bandwidth x_{ij} will result in a corresponding increment (decrement) of the bandwidth b_{ij} obtained by the scheduler if the system is not in overload.

We now describe in more detail how the operation of slot negotiation works for the agent a_{ij} associated with input/output pair (i, j) and with queue q_{ij}. We describe the agent behavior at the beginning of a generic frame; thus, we do not explicitly indicate the frame index in the notation. In the sequel, when needed, we will use the notation $[n]$ to indicate that frame with index n is considered.

Each agent bases its decision on a cost function, which is minimized by each agent to determine which is the optimal amount of bandwidth to negotiate. The cost function C_{ij} for agent a_{ij} is composed by adding a performance cost function Q and a bandwidth cost function P:

$$C_{ij} = Q(b_{ij}) + P(x_{ij}, I_i, O_j) \qquad (1)$$

where $I_i = \sum_j x_{ij}$ is the total amount of bandwidth purchased from input i, and $O_j = \sum_i x_{ij}$ is the total amount of bandwidth purchased towards output j.

The term $Q(b_{ij})$ of Eq. (1) corresponds to a *performance cost* (penalty), which is a function of the assigned bandwidth b_{ij}. If the performance penalty increases, the agent will be induced to buy bandwidth, in order to improve his performance. Otherwise, if $Q(b_{ij})$ decreases, the agent interests will be to sell bandwidth, in order to decrease his final cost.

The term $P(x_{ij}, I_i, O_j)$ of Eq. (1) represents the *bandwidth cost*, that is, the cost of the purchased bandwidth x_{ij}. It is computed based upon information sent by the scheduler to the agents (I_i and O_j can be not locally known). This term has the goal of indirectly inducing the agents to play in a coordinated fashion: to obtain this goal, the function is set growing to infinity when the total purchased bandwidth for an input or an output is close to the saturation. In other words, this function is infinity when $\max(I_i, O_j) \to 1$. We will discuss in Sect. V-A all the exact conditions that functions $Q(\cdot)$ and $P(\cdot)$ must satisfy to end up with an efficient and stable SSB game.

Let us clarify the relation between the cost function and x_{ij}. Let $z_{ij}[n]$ be the number of slots negotiated by, and assigned to agent a_{ij} in frame n; clearly, $x_{ij}[n] = z_{ij}[n]/F$. During the operation of slot purchase, $z_{ij}[n]$ is computed according to the following simple "differential" algorithm:

$$z_{ij}[n] = \min\{F, \max\{0, z_{ij}[n-1] + \Delta_{ij}[n]\}\}$$

where $\Delta_{ij}[n]$ can assume two values $\{-1, +1\}$ (corresponding to actions of slot "selling" and "buying"), according to the following algorithm:

$$\Delta_{ij}[n] = \begin{cases} +1 & \text{if } \lambda_{ij} \geq b_{ij} \;\texttt{AND}\; \lambda_{ij} > 0 \\ +1 & \text{if } \lambda_{ij} < b_{ij} \;\texttt{AND}\; \frac{\partial C_{ij}}{\partial x_{ij}} < 0 \\ -1 & \text{if } \lambda_{ij} = 0 \\ -1 & \text{if } \lambda_{ij} < b_{ij} \;\texttt{AND}\; \frac{\partial C_{ij}}{\partial x_{ij}} \geq 0 \end{cases}$$

We choose to permit only one-slot variations per frame for simplicity. If we allow Δ to assume values up to $\pm h$, the allocated bandwidth may change up to h/F during each frame, for each input/output pair. This influences the reactivity of the system to traffic changes: h should be tailored on the basis of the maximum allowed traffic changes. However, the important system parameter is the ratio h/F; fixing $h = 1$ imposes some constraint on the choice of F, but does not prevent the selection of a correct time constant for the system in operation. We do not tackle this issue deeply in this paper, whose aim is to describe a flexible framework for dealing with frame based scheduling, rather than optimizing system parameters.

Note that the above scheme, being based on independent choices made by each agent, may create an overload in overall slot requests; this is a transient phenomenon, given the assumption that the system is not in overload. A simple ceiling scheme to make slot requests feasible is used in slot allocation. Fairness is ensured by a round robin selection among agents.

For the computation of the bandwidth allocation matrix in the bandwidth computation phase, we consider the following rule:

$$b_{ij}(X) = x_{ij} + \frac{1}{N}(1 - \max(I_i, O_j)) \qquad (2)$$

In other words, the excess bandwidth is evenly shared among competing flows. With this scheme, the maximum bandwidth variation per frame for each input (or output) is N slots. This constrains the worst case complexity of the plan-scheduling phase, which is differential in nature.

We consider now some realistic cost functions that could be reasonably implemented in real switches.

Performance cost:

- *Average queue length*: $Q(b_{ij}) = \lambda_{ij}/(b_{ij} - \lambda_{ij})$ (derived from the average length of an $M/D/1$ queue) for $b_{ij} > \lambda_{ij}$, and $Q[b_{ij}] = +\infty$ when $b_{ij} \leq \lambda_{ij}$. $Q(b_{ij})$ depends on the average arrival rate λ_{ij}: if λ_{ij} is not known a priori, an on-line estimator of λ_{ij} is required. This function satisfies the conditions of Sect. V-A for the existence and uniqueness of a NEP.
- *Average delay*: $Q(b_{ij}) = 1/(b_{ij} - \lambda_{ij})$ (derived from the average delay of an $M/D/1$ queue) for $b_{ij} > \lambda_{ij}$, and $Q(b_{ij}) = \infty$ when $b_{ij} \leq \lambda_{ij}$. Also in this case, if λ_{ij} is not a priori known, an on-line estimator of λ_{ij} is required. This function satisfies the conditions of Sect. V-A for the existence and uniqueness of a NEP.
- *Queue length*: $\partial Q_{ij}/\partial b_{ij} = -L_{ij}[n]$; in this case, the variation of the performance cost function is evaluated by just looking at the instantaneous queue length at time n; the main advantage of this metric is that it does not require any knowledge or estimation for λ_{ij}; thus, it is an appealing choice for practical implementations. The intuitive explanation is that, as the queue grows, the incentive for the agent to buy more bandwidth increases, since the decrease in the cost function is more sensible to the bandwidth. More formally, consider the following performance cost, after setting small $\epsilon > 0$:

$$Q_{ij}(b_{ij}) = -\int_{\lambda_{ij}+\epsilon}^{b_{ij}} E[L_{ij}(\theta)]d\theta + \int_{\lambda_{ij}+\epsilon}^{1} E[L_{ij}(\theta)]d\theta$$

where $E[L_{ij}(b_{ij})]$ is the generic relation between the average queue length and the service rate. Since $b_{ij} \leq$

1, then Q_{ij} is always finite and positive. Hence, $\partial Q_{ij}/\partial b_{ij} = -E[L_{ij}(b_{ij})]$, that is Q_{ij} is a decreasing function of b_{ij}. Now we can check the convexity of the function: $\partial^2 Q_{ij}/\partial b_{ij}^2 = -\partial E[L_{ij}]/\partial b_{ij} > 0$, since for any $M/G/1$ queuing system the average queue length is decreasing with the service rate. We then approximate $\partial Q_{ij}/\partial b_{ij} = -E[L_{ij}(b_{ij})]$ with the current $-L_{ij}[n]$. The main disadvantage of this rough approximation is that with this cost function we cannot prove the same stability properties of the other performance cost functions, as done in Sect. V-D. Nevertheless, in Sect. VI we show by simulation that the system using this function performs very similarly to the system based on other, theoretically stable, functions.

Bandwidth cost:

- *Proportional bandwidth cost*: $P(\cdot) = x_{ij}/(1 - \max(I_i, O_j))$, being $1/(1 - \max(I_i, O_j))$ the "cost per unit of bandwidth".
- *Fixed bandwidth cost*: $P(\cdot) = 1/(1 - \max(I_i, O_j))$.

Several combinations of performance and bandwidth cost functions can be envisioned. For example, if we set the cost function C_{ij} to the sum of the two following terms:

$$Q(b_{ij}) = \frac{\lambda_{ij}}{(b_{ij} - \lambda_{ij})},\ P(x_{ij}, I_i, O_j) = \frac{x_{ij}}{1 - \max\{I_i, O_j\}}$$

when $b_{ij} > \lambda_{ij}$, $\Delta[n]$ is computed on the basis of:

$$\frac{\partial C_{ij}}{\partial x_{ij}} = \frac{\partial Q}{\partial b_{ij}}\frac{\partial b_{ij}}{\partial x_{ij}} + \frac{\partial P}{\partial x_{ij}} =$$
$$= \frac{-\lambda_{ij}}{(b_{ij} - \lambda_{ij})^2}\left(1 - \frac{1}{N}\right) + \frac{1 + x_{ij} - \max\{I_i, O_j\}}{(1 - \max\{I_i, O_j\})^2}$$

When the arrival rates $\{\lambda_{ij}\}$ are unknown, some estimators $\{\hat{\lambda}_{ij}\}$ have to be employed, tailored to the traffic characteristics. For example, if the traffic is stationary, then a simple integrator can be used:

$$\hat{\lambda}_{ij}[n] = \frac{1}{n}\sum_{t=1}^{n} a_{ij}[t]$$

being $a_{ij}[t]$ equal to 1 when an arrival is experienced at queue q_{ij}. Otherwise, if the traffic is time-variant, then a first order filter can be used:

$$\hat{\lambda}_{ij}[n] = \alpha\hat{\lambda}_{ij}[n-1] + (1-\alpha)a_{ij}[n] \tag{3}$$

where α, $0 < \alpha < 1$, determines the filter reactivity: $1/(1-\alpha)$ is the approximate number of frames necessary to correctly estimate a new arrival rate.

B. Plan-Scheduling Phase

The classic Paull algorithm is used to update the scheduling plan, on the basis of $B[n]-B[n-1]$, i.e. to increase or decrease the number of slots given in the frame for each input/output pair. Paull algorithm was defined to configure rearrangeable circuit switching three-stage Clos interconnection networks. The equivalence between frame scheduling and the configuration of Clos networks has been deeply discussed in [15]. Paull algorithm, as described in [16], may be simply adapted to frame scheduling by looking at the identifier of a middle-stage switching module in a Clos network as time slot position inside the frame, and at each first/third stage switching module as an input/output port of an IQ switch. The most interesting feature of the Paull algorithm is that it can increase and decrease the bandwidth allocated in a frame with the granularity of a single slot.

Using the efficient data structure (called "Paull-matrix") proposed in [16], the computational complexity can be estimated as $O(N)$ for each slot addition, and $O(1)$ for each slot removal. Paper [15] discusses the implementation details of the algorithm. Note that the complexity is independent from the frame size F. Paull algorithm can be parallelized and be implemented very efficiently in hardware, following the design ideas of controllers for traditional Clos networks.

V. The game theoretical model of the Scheduling Policy

In this section we will apply basic concepts from game theory in order to prove properties of our SSB policy.

A. Existence of Nash equilibria

In the following, we neglect the quantization effect of the bandwidth that each agent can purchase, due to the finiteness of the frame size F.

Here we briefly summarize the main properties of above described functions, $C_{ij}(x_{ij}, I_i, O_j)$, $Q(b_{ij})$, $P(x_{ij}, I_i, O_j)$, $b_{ij}(x_{ij}, I_i, O_j)$, which we will exploit in the following proofs[1].

- Total cost: $C_{ij}(x_{ij}, I_i, O_j) = Q(b_{ij}) + P(x_{ij}, I_i, O_j)$.
- Performance cost:
  1) $Q(b_{ij}) = +\infty$ for $b_{ij} \le \lambda_{ij}$.
  2) $Q(b_{ij})$ is continuous and double derivable, strictly decreasing and strictly convex[2] function with respect to b_{ij}, where it is finite.
- Bandwidth cost:
  1) $P(x_{ij}, I_i, O_j)$ is continuous and doubly derivable, increasing and strictly convex function with respect to x_{ij}.
  2) $P(x_{ij}, I_i, O_j)$ is weakly increasing and weakly convex with respect to I_i and O_j.
  3) $\lim_{x_{ij}\to 1} P(x_{ij}, I_i, O_j) = \lim_{I_i \to 1} P(x_{ij}, I_i, O_j)$ $= \lim_{O_j \to 1} P(x_{ij}, I_i, O_j) = +\infty$.
- Bandwidth allocation:
  1) $b_{ij}(x_{ij}, I_i, O_j) = x_{ij} + (1 - \max(I_i, O_j))/N$ is a strictly-increasing function, weakly-concave with respect to x_{ij}, satisfying the constraint $b_{ij} \ge x_{ij}$;

[1] In the sake of readability, we will omit some variables inside the $(\cdot)$ when they are clear from the context.

[2] $f(x)$ is weakly convex if $\partial^2 f/\partial x^2 \ge 0$; $f(x)$ is strictly convex if $\partial^2 f/\partial x^2 > 0$.

in addition[3], $\partial b_{ij}/\partial x_{ij} = 1$ and $db_{ij}/dx_{ij} = (1 - 1/N)$.

2) $b_{ij}(x_{ij}, I_i, O_j)$ is decreasing and weakly-concave with respect to $\max(I_i, O_j)$; $\partial b_{ij}/\partial \max(I_i, O_j) = -1/N$.

All of the previous properties are of immediate verification in our scheme.

Theorem 1: For the *basic* SSB game, in which the strategy space is:

$$A_X = \{X : I_i \leq 1, O_j \leq 1, b_{ij}(X) \geq \lambda_{ij}, \forall i, \forall j\}$$

at least a finite Nash Equilibrium Point (NEP) exists.

Proof: $Q(b_{ij}(x_{ij}, I_i, O_j))$ can be immediately verified to be weakly-convex with respect to x_{ij}, since $Q(b_{ij})$ is convex, strictly-decreasing and $b_{ij}(x_{ij}, I_i, O_j)$ is concave. Since also $P(x_{ij}, I_i, O_j)$ is convex with respect to x_{ij}, then $C_{ij}(x_{ij}, I_i, O_j)$ is convex with respect to x_{ij}. Now, if we prove that A_X is compact and convex, the existence of a NEP is guaranteed by the Kakutani fixed point theorem [17].

The compactness of A_X derives immediately from the fact that it is a closed and bounded subset of $\mathbb{R}^{N^2}$. To apply the Kakutani theorem, it is only necessary to show that A_X is convex, i.e., for any couple of matrices $X^{(1)}$ and $X^{(2)}$ both belonging to A_X, all matrices $X^{(\alpha)} = \alpha X^{(1)} + (1-\alpha) X^{(2)}$ for $\alpha \in [0, 1]$ lie in A_X too. Let $B(X)$ be the instantaneous bandwidth matrix corresponding to X. For any agent a_{ij}, it holds:

$$\begin{aligned}
&b_{ij}(x_{ij}^{(\alpha)}, \max(I_i^{(\alpha)}, O_j^{(\alpha)})) = \\
&\quad = \alpha b_{ij}(x_{ij}^{(1)}, \max(I_i^{(1)}, O_j^{(1)})) + (1-\alpha) b_{ij}(x_{ij}^{(2)}, \\
&\qquad \max(I_i^{(2)}, O_j^{(2)})) \geq \alpha \lambda_{ij} + (1-\alpha)\lambda_{ij} = \lambda_{ij}
\end{aligned}$$

Thus it results $X^{(\alpha)} \in A_X$. ■

Corollary 1: For the basic SSB game, all the possible NEPs are finite. In other words, if X^* is a NEP with corresponding B^*, then: $b_{ij}^* > \lambda_{ij}$ and $\max(I_i^*, O_j^*) < 1$.

Proof: Assume by contradiction that an infinite NEP X^* exists and the corresponding bandwidth is B^*; then two cases are possible: (i) $\max(I_i^*, O_j^*) = 1$, or (ii) $b_{ij}^* = \lambda_{ij}$, for some agent a_{ij}. In case (i), say that $I_i^* = 1$; then all agents a_{ik} experience an infinite bandwidth cost; among them there is at least an agent a_{il} whose performance cost $Q(x_{il}^*)$ is not infinite since the traffic is admissible. Thus agent a_{il} would decrease his own cost by selling bandwidth, and X^* cannot be a NEP. The same argument holds in the case $O_j^* = 1$. In case (ii), agent a_{ij} experiences an infinite performance cost $Q(x_{ij}^*)$, then it would reduce his own cost by buying some extra bandwidth (whose cost is finite since $\max(I_i^*, O_j^*) < 1$), and X^* cannot be a NEP. ■

In the *extended* SSB game, the strategy space A'_X include also vectors X for which the constraint $b_{ij} \geq \lambda_{ij}$ is violated: $A'_X = \{X : I_i \leq 1, O_j \leq 1, \forall i, \forall j\}$, i.e. some players can experience an infinite cost function. For this extended switching game, the agent should deal with infinite values of the cost function. We say that

$$C_{ij} = Q(b_{ij}) + P(x_{ij}, I_i, O_j) \qquad \text{for} \quad b_{ij} > \lambda_{ij}$$

However, for values $b_{ij} \leq \lambda_{ij}$ to which an infinite cost function corresponds, we postulate that greater values of x_{ij} are strictly preferred by agent a_{ij}, since they correspond to larger values of throughput b_{ij}.

Theorem 2: All the possible NEPs for the basic SSB game are also NEPs for the extended SSB game.

Proof: Consider a point X^* which is a (finite) NEP for the basic game. Say that agent a_{ij} of the extended game has bought x^* and $\tilde{x}^*$ is the vector of all the other elements of X^*. Consider now the strategy space A_x on agent a_{ij} conditioned on the other agents' purchased bandwidths $\tilde{x}^*$, i.e. A_x is the restriction of A_X on a segment; note that the minimum value of A_x is the x' such that $b(x', \tilde{x}^*) = \lambda_{ij}$. Because of Corollary 1, x^* is an internal point of A_x and $C_{ij}(x, \tilde{x}^*)$ is convex with respect to x where it is finite. Hence, x^* is a local minimum of $C_{ij}(x, \tilde{x}^*)$. Now consider A'_x defined as the restriction of A'_X on the strategy space of agent a_{ij}. Note that $A_x \subseteq A'_x$ and $x = x^*$ is still a local minimum of $C_{ij}(x, \tilde{x}^*)$, that is a global minimum, because C_{ij} is still convex on A'_x where it is finite. Then, the NEP will still remain in X^*. ■

By following the same reasoning of the proof of Corollary 1, it can be shown:

Corollary 2: For the extended SSB game, all the possible NEPs are finite.

B. NEP uniqueness

Theorem 3: The extended SSB game admits a unique NEP.

Proof: By contradiction, assume that two different NEPs $X^{(1)}$ and $X^{(2)}$ exists. Consider an agent a_{ij}. We can have four different cases.

Case (a). If $\max(I_i^{(1)}, O_j^{(1)}) \leq \max(I_i^{(2)}, O_j^{(2)})$ then $b_{ij}^{(1)} \geq b_{ij}^{(2)}$. To show this case, we will prove that either the cases (a.1) $b_{ij}^{(1)} < b_{ij}^{(2)}$ and $x_{ij}^{(1)} \geq x_{ij}^{(2)}$, or (a.2) $b_{ij}^{(1)} < b_{ij}^{(2)}$ and $x_{ij}^{(1)} < x_{ij}^{(2)}$, are impossible.

Case (a.1). Suppose $b_{ij}^{(1)} < b_{ij}^{(2)}$ and $x_{ij}^{(1)} \geq x_{ij}^{(2)}$. b_{ij} is increasing with respect to x_{ij} and decreasing with respect to $\max(I_i, O_j)$. Thus $b_{ij}^{(1)} = b(x_{ij}^{(1)}, \max(I_i^{(1)}, O_j^{(1)})) \geq b(x_{ij}^{(2)}, \max(I_i^{(1)}, O_j^{(1)})) \geq b(x_{ij}^{(2)}, \max(I_i^{(2)}, O_j^{(2)})) = b_{ij}^{(2)}$ which is in contrast with the assumptions.

Case (a.2). Suppose $b_{ij}^{(1)} < b_{ij}^{(2)}$ and $x_{ij}^{(1)} < x_{ij}^{(2)}$. At both NEPs the Kuhn-Tucker conditions must be satisfied:

$$\left.\frac{\partial C_{ij}(X)}{\partial x_{ij}}\right|_{X^{(1)}} = \left.\frac{\partial C_{ij}(X)}{\partial x_{ij}}\right|_{X^{(2)}} = 0 \qquad \forall i, j \tag{4}$$

with

$$\frac{\partial C_{ij}(X)}{\partial x_{ij}} = \frac{\partial Q(b_{ij})}{\partial b_{ij}} \frac{\partial b(x_{ij}, \max(I_i, O_j))}{\partial x_{ij}} + \\ + \frac{\partial P(x_{ij}, I_i, O_j)}{\partial x_{ij}}$$

[3] In the notation, we distinguish between partial and total derivatives, i.e. $df(x, y(x))/dx = \partial f(x, y)/\partial x + \partial f(x, y)/\partial y \times \partial y/\partial x$.

Due to the definition of $b_{ij}(X)$, $\partial b_{ij}/\partial x_{ij}$ is a positive constant and $Q_{ij}(b_{ij})$ is strictly convex and decreasing:

$$\left.\frac{\partial Q_{ij}}{\partial x_{ij}}\right|_{X^{(1)}} = \left.\frac{\partial Q_{ij}}{\partial b_{ij}}\right|_{B^{(1)}} \left.\frac{\partial b_{ij}}{\partial x_{ij}}\right|_{X^{(1)}} <$$

$$< \left.\frac{\partial Q_{ij}}{\partial b_{ij}}\right|_{B^{(2)}} \left.\frac{\partial b_{ij}}{\partial x_{ij}}\right|_{X^{(2)}} = \left.\frac{\partial Q_{ij}}{\partial x_{ij}}\right|_{X^{(2)}}$$

To satisfy the Kuhn-Tucker conditions (4), it must be:

$$\left.\frac{\partial P_{ij}}{\partial x_{ij}}\right|_{X^{(1)}} > \left.\frac{\partial P_{ij}}{\partial x_{ij}}\right|_{X^{(2)}} \tag{5}$$

But the last relation is impossible since $P(x_{ij}, I_i, O_j)$ is convex in both x_{ij} and $\max(I_i, O_j)$.

As a conclusion of previous arguments, we proved that: (a) $\max(I_i^{(1)}, O_j^{(1)}) \leq \max(I_i^{(2)}, O_j^{(2)})$ necessarily entails $b_{ij}^{(1)} \geq b_{ij}^{(2)}$.

Case (b). If $\max(I_i^{(1)}, O_j^{(1)}) \geq \max(I_i^{(2)}, O_j^{(2)})$, then $b_{ij}^{(1)} \leq b_{ij}^{(2)}$. Indeed, this case can be shown by exchanging the roles between $X^{(1)}$ and $X^{(2)}$ in case (a).

Case (c). If $\max(I_i^{(1)}, O_j^{(1)}) < \max(I_i^{(2)}, O_j^{(2)})$ and $b_{ij}^{(1)} = b_{ij}^{(2)}$ then $x_{ij}^{(1)} < x_{ij}^{(2)}$. Indeed, since $b_{ij}(x_{ij}, \max(I_i, O_j))$ is strictly increasing in the first argument, and strictly decreasing in the second argument.

Case (d). If $\max(I_i^{(1)}, O_j^{(1)}) = \max(I_i^{(2)}, O_j^{(2)})$ then $x_{ij}^{(1)} = x_{ij}^{(2)}$. Indeed, if we assume $x_{ij}^{(1)} < x_{ij}^{(2)}$, then $b_{ij}^{(1)} < b_{ij}^{(2)}$, which is in contradiction with case (a). If we instead assume $x_{ij}^{(1)} > x_{ij}^{(2)}$ then would result $b_{ij}^{(1)} > b_{ij}^{(2)}$, thus in contradiction with case (b).

Now we show that if, $X^{(1)}$ and $X^{(2)}$ are different, then there exists at least an agent a_{ij} for which (a), (b), (c) or (d) are necessarily violated.

Consider now when $\max_i \max(I_i^{(1)}, O_i^{(1)}) \neq \max_i \max(I_i^{(2)}, O_i^{(2)})$; without loss of generality, assume: $\max_i \max(I_i^{(1)}, O_i^{(1)}) < \max_i \max(I_i^{(2)}, O_i^{(2)}) = \alpha$ and $I_0^{(2)} = \alpha$. Then consider the elements on the first row of $X^{(1)}$ and $X^{(2)}$; from $b_{ij} \geq x_{ij}$, it results $\sum_j b_{0j}^{(2)} = 1$, since $I_0^{(2)} = \max(I_0^{(2)}, O_j^{(2)})$, while in general $\sum_j b_{0j}^{(1)} \leq 1$, since we are not sure that $I_0^{(1)} = \max(I_0^{(1)}, O_j^{(1)})$. Then there are two cases:

Case (i). $b_{0j}^{(1)} = b_{0j}^{(2)}$ for all j; then by (c) $x_{0j}^{(1)} < x_{0j}^{(2)}$ for any j. At the same time,

$$\left.\frac{\partial Q_{0j}}{\partial b_{0j}}\right|_{B^{(1)}} = \left.\frac{\partial Q_{0j}}{\partial b_{0j}}\right|_{B^{(2)}} \qquad \forall j$$

Being $\partial b_{0j}/\partial x_{0j}$ a constant, to satisfy the Kuhn-Tucker conditions, it should be:

$$\left.\frac{\partial P_{0j}}{\partial x_{0j}}\right|_{X^{(1)}} = \left.\frac{\partial P_{0j}}{\partial x_{0j}}\right|_{X^{(2)}} \qquad \forall j$$

But $P(\cdot)$ is strictly convex with respect to x_{ij} and $\max(I_i, O_j)$; since, $x_{0j}^{(1)} < x_{0j}^{(2)}$ and $\max(I_0^{(1)}, O_j^{(1)}) < \max(I_0^{(2)}, O_j^{(2)})$ for all j, then the final result is a contradiction:

$$\left.\frac{\partial P_{0j}}{\partial x_{0j}}\right|_{X^{(1)}} < \left.\frac{\partial P_{0j}}{\partial x_{0j}}\right|_{X^{(2)}} \qquad \forall j$$

Case (ii). There exist some agents a_{0j} such that $b_{0j}^{(1)} \neq b_{0j}^{(2)}$, but in this case at least for one j it must hold $b_{0j}^{(1)} < b_{0j}^{(2)}$ (otherwise it cannot be $1 = \sum_j b_{0j}^{(2)} \geq \sum_j b_{0j}^{(1)}$). Thus (a) is violated.

Now consider the case in which $\max_i \max(I_i^{(1)}, O_i^{(1)}) = \max_i \max(I_i^{(2)}, O_i^{(2)}) = \alpha$, if there exists an input i (or an output j) such that $I_i^{(2)} = \alpha$ and $I_i^{(1)} < \alpha$ (or $O_j^{(2)} = \alpha$ and $O_j^{(1)} < \alpha$), we can apply the same reasoning as for the previous case to show that a contradiction necessarily arises.

Otherwise for any maximal row i (or maximal column j), it must be $x_{ik}^{(1)} = x_{ik}^{(2)}$, $b_{ik}^{(1)} = b_{ik}^{(2)}$, and $\forall k$, ($x_{kj}^{(1)} = x_{kj}^{(2)}$, and $b_{kj}^{(1)} = b_{kj}^{(2)}$ $\forall k$), from (d). In this case we can neglect maximal rows (columns) and repeat the previous arguments to the sub-matrices $X_*^{(1)}$ and $X_*^{(2)}$ containing the elements of $X^{(1)}$ and $X^{(2)}$ respectively, which are not on maximal rows (columns). Iterating the reasoning, it results that the contradictions are avoided only if $X^{(1)} = X^{(2)}$. ■

C. Convergence to the NEP

In this section we discuss the convergence properties of the iterative algorithms to the unique NEP. We consider the scenario in which periodically all agents at the same time update the purchased bandwidth, according to the following the Jacobi iterative scheme: $x_{ij}[n+1] =$

$$= \begin{cases} x_{ij}[n] - \alpha \frac{\partial C_{ij}}{\partial x_{ij}}[n] & \text{if } \begin{cases} \max(I_i[n], O_j[n]) < 1 \\ \texttt{AND } b_{ij}[n] > \lambda_{ij} \end{cases} \\ \max(0, x_{ij}[n] - \delta) & \text{if } \max(I_i[n], O_j[n]) = 1 \\ \min(1, x_{ij}[n] + \delta) & \text{if } \begin{cases} \max(I_i[n], O_j[n]) < 1 \\ \texttt{AND } b_{ij}[n] \leq \lambda_{ij} \end{cases} \end{cases}$$

where $\alpha > 0$ and $0 < \delta < 1$ are two parameters of the algorithm.

Theorem 4: The extended SSB game, adopting Jacobi iterative scheme, converges to the unique NEP for a small enough α.

Proof: We only sketch the proof. Let X^* be the only NEP to which the matrix B^* of instantaneous bandwidth corresponds. Let us consider $X[n]$, the algorithm operational point at time-instant n. We suppose that there exists an $\epsilon > 0$ such that both $b_{ij}[n] > \lambda_{ij} + \epsilon$ and $\sum x_{ij}[n] < 1 - \epsilon$. In this case we can suppose that $\partial C_{ij}/\partial x_{ij}$ is bounded, i.e., there exists a finite M such that $|\frac{\partial C_{ij}}{x_{ij}}| < M$. This assumption can be relaxed by observing that in all operational points that do not satisfy the previous assumptions the cost function is unbounded. As a consequence, if the system starts from a point that does not satisfy the previous assumptions, it will move toward the region that satisfies the previous assumptions, from which it will never escape if α is sufficiently small.

Under previous assumptions it is immediate to prove that: (a) if $b_{ij}[n] \geq b_{ij}^*$ along with $\max(I_i[n], O_j[n]) \geq \max(I_i^*, O_j^*)$, then $x_{ij}[n] \geq x_{ij}^*$ (as already shown in the proof of Theorem 3) and $\frac{\partial C_{ij}}{\partial x_{ij}} \geq 0$; in a similar way, (b) if $b_{ij}[n] \leq b_{ij}^*$ along with $\max(I_i[n], O_j[n]) \leq \max(I_i^*, O_j^*)$, then $x_{ij}[n] \leq x_{ij}^*$ and $\frac{\partial C_{ij}}{\partial x_{ij}} \leq 0$.

Note that, if $X[n] \neq X^*$, at least one agent a_{ij} exists such that at the same time $x_{ij}[n] - x_{ij}^* > 0$ and $\max(I_i[n], O_j[n]) -$

$\max(I_i^*, O_j^*) \geq 0$ (or $x_{ij}[n]-x_{ij}^* < 0$ and $\max(I_i[n], O_j[n]) - \max(I_i^*, O_j^*) \leq 0$).

Thus let us define the following functional of $X[n]$:

$$V(X[n]) = \max\{\max_{ij}(x_{ij}[n] - x_{ij}^*) \times$$
$$\times\, \mathbb{1}[\max(I_i[n], O_j[n]) - \max(I_i^*, O_j^*)], \max_{ij}(x_{ij}^* - x_{ij}[n]) \times$$
$$\times\, \mathbb{1}[\max(I_i^*, O_j^*) - \max(I_i[n], O_j[n])]\}$$

where $\mathbb{1}[x] = 1$ if $x \geq 0$, and $\mathbb{1}[x] = 0$ if $x < 0$.

Since $V(X[n]) \geq 0$ and $V(X[n]) = 0$ only if $X[n] = X^*$, $V(X[n])$ can be interpreted as a Lyapunov function for the system. Moreover it can be proved that, for each element (i', j') that maximizes $x_{ij}[n] - x_{ij}^* > 0$ on the matrix rows and columns (or minimizes $x_{ij}[n] - x_{ij}^* < 0$), then it results $b_{i'j'}[n] - b_{i'j'}^* \geq 0$ (or $b_{i'j'}[n] - b_{i'j'}^*) \leq 0$).

Thus, for all the elements belonging to $\arg\max_{ij}(x_{ij}[n] - x_{ij}^*) \times \mathbb{1}[\max(I_i[n], O_j[n]) - \max(I_i^*, O_j^*)]$, either (a) or (b) must hold. As a consequence, for small values of α, the functional $V(X[n])$ defined below is non increasing with respect to n, which implies the equilibrium point X^* to be stable (i.e., the algorithm converges to the NEP). ■

D. Maximum achievable throughput

We now prove the stability properties of the SSB game.

Definition 1: A switching architecture achieves 100% throughput, or it is *rate stable*, if

$$\lim_{n\to\infty} \frac{L_{ij}[n]}{n} = 0 \quad \text{w.p.1} \qquad \forall i, j$$

where $L_{ij}[n]$ is the queue length at the beginning of frame n.

Consider now the case of performance costs given by the average queue length or the average delay (as defined in Sec. IV-A). We start to assume an ideal knowledge of the average traffic matrix Λ, which is admissible.

Theorem 5: Under any admissible traffic pattern that satisfies the strong law of large numbers, an input-queued switch implementing the extended SSB policy asymptotically achieves 100% throughput if the frame dimension F is sufficiently large.

Proof: Since $b_{ij} \geq \lambda_{ij}$ at the NEP, the switching system is rate stable, as formally proved in ([13], Theorem 1). ■

The general assumption that the arrival rates are known is not practical. When the rates are unknown, a practical scheme can consider in $Q(b_{ij})$ some rate estimators $\hat{\lambda}_{ij}$ instead of λ_{ij}. Luckily, the stability result continues to hold:

Theorem 6: Under any admissible traffic pattern that satisfies the strong law of large numbers, an input-queued switch implementing the extended SSB policy with online rate estimation asymptotically achieves 100% throughput, if the frame dimension F is sufficiently large, and the rate estimation converges to the true arrival rate.

Proof: Under the large F assumption, we can neglect the quantization effect. In addition, the traffic estimations $\hat{\lambda}_{ij}[n]$ tends w.p.1 to λ_{ij} for $n \to \infty$; thus for each $\epsilon > 0$, there exists an n_0 such that $|\hat{\lambda}_{ij}[n] - \lambda_{ij}| < \epsilon$ for any $n > n_0$ with probability one. But since the iterative algorithm converges, as proved before, to an efficient NEP at which $b_{ij}^* > \hat{\lambda}_{ij}[n]$ there exists a time instant n_1 such that $b_{ij}[n] > \hat{\lambda}_{ij}[n] + \delta$, for some $\delta > 0$ and any $n > n_1$. Being ϵ arbitrary, we can always set $\epsilon < \delta$ thus resulting: $b_{ij}[n] > \lambda_{ij}[n]$, for any $n > n_1$, i.e. the bandwidth provided to agent a_{ij} is greater than the traffic arrival rate, which implies the rate stability of the switch as formally proved in ([13], Theorem 1). ■

Theorem 7: Under any admissible traffic pattern that satisfies the strong law of large numbers, a *network* of input-queued switches all implementing the SSB policy asymptotically achieves 100% throughput if the frame dimension F is sufficiently large and the rate estimation referred to entering flows converges to the true arrival rate.

Proof: The proof immediately comes from the observation that, if we are able to prove that the traffic estimations $\hat{\lambda}_{ij}[n] \to \lambda_{ij}$ w.p.1 as $n \to \infty$, then repeating the same argument of the previous proof, we can show that the bandwidth provided at each switch to every input-output port traffic relation is greater than the corresponding average arrival rate, which implies the rate stability of switch networks, as formally proved in ([13], Theorem 1). To prove that $\hat{\lambda}_{ij}[n]$ tends w.p.1 to λ_{ij} for $n \to \infty$, we can exploit the same arguments of ([13], Theorem 2). ■

VI. Simulation Results

In this section, we analyze by simulation the merits of the proposed SSB policy under several possible combinations of performance cost and bandwidth cost functions. When the knowledge of the arrival rates λ_{ij} is required by the cost function, we use the filter estimators $\hat{\lambda}_{ij}$ given by (3). We test the SSB policy under i.i.d. Bernoulli arrivals, with both stationary and time-variant traffic, distributed according to a given traffic matrix.

We consider a switch with $N = 32$ input and output ports, each running at 10 Gbps. We assume a cell size of 64 bytes, that is 512 bits, a common internal cell format used in switching fabrics. Hence, the slot duration is 51.2 ns. We set the size of each VOQ equal to 2000 cells, that is 128 KBytes for each VOQ, and 4 MBytes for the whole input port. The normalized input load ρ ranges between 0.2 and 0.99, equivalent to data rates between 2 and 9.9 Gbps. We present here only results obtained for frame size $F = 128$ and 1024 cell slots, equivalent to a frame duration of 6.5 μs and 52 μs respectively, which we consider good representatives of short and long frame sizes, although several other values of F were considered.

We define the following traffic matrices:

- *Uniform traffic*: $\lambda_{ij} = \rho/N$. This is the most classical scenario considered in the literature. Almost all the proposed algorithms in the literature achieve the maximum throughput under this traffic.
- *Log-diagonal traffic*: define $\bar{\lambda}_d = \rho(2^{N-1-d})/(2^N - 1)$, with $d = 0, \ldots, N-1$ representing a matrix diagonal index[4], i.e., $a_{i|i+d|_N} = a_{ij}$; then, $\lambda_{ij} = \bar{\lambda}_d$. In this case the load halves from one diagonal to the other. This traffic

[4] In the following $|.|_N$ is the modulo-N operator.

matrix is usually very critical to be scheduled, and most of the non-optimal algorithms are not able to achieve maximum throughput.

- *Lin-diagonal traffic*: $\bar{\lambda}_d = \rho(N-d)/(N(N+1)/2)$, with $d = 0, \dots, N-1$, then $\lambda_{ij} = \bar{\lambda}_d$ if $j = |i+d|_N$; the load decreases linearly from one diagonal to the other. This traffic matrix is an intermediate case between the uniform and log-diagonal ones.

In the following subsections, we show that (i) no throughput limitation is observed even when the cost function does not theoretically guarantee that an equilibrium is reached, (ii) good performance in terms of delay are obtained in most scenarios, even if a delay penalty should be paid with respect to optimal slot by slot schedulers, (iii) the proposed framework is very flexible, and permits to obtain differentiated delays by a proper setting of parameters in the cost function.

A. Performance under stationary traffic

To give insights into the system behavior, before examining performance results, we introduce an example of agent behavior in a switch loaded with a log-diagonal matrix. We discuss the temporal evolution of the number of purchased slots during a transient period, i.e., starting from a slot allocation not matched to the log-diagonal traffic: we assume, at simulation startup, that the number of purchased slots are null for all queues, i.e. $x_{ij} = 0, \forall i, j$. For simplicity, we show and describe the behaviour of only 5 among the 32 agents at input port 0: agents playing the game for outputs $0-4$, i.e. $a_{ij}, i = 0, j = 0, 1, 2, 3, 4$. Since the considered traffic is log-diagonal, the initial portion of the vector $\bar{\Lambda}$ of arrival rates at input 0, where $\bar{\lambda}_j$ is the traffic toward output j, is $[0.45, 0.225, 0.112, 0.056, 0.028]$. Being $\rho = 0.9$ and $F = 256$, the number of purchased slots by each agent at the NEP is very close to $F\lambda_j$, whose corresponding vector is $[116, 58, 29, 15, 8]$. Thus, the rightmost part of the graph in Fig. 1 shows that the SSB policy achieves the NEP, given that the number of purchased slots takes values close to those derived above. Looking at the leftmost part of the graph in Fig. 1, when the SSB game starts, all the agents have bought 0 slots in the frame; hence the received bandwidth is $1/N = 0.031$ for all the agents. This bandwidth is not sufficient to satisfy the first 4 agents, since their arrival rate is greater than the service rate; thus, the agents keep buying slots. The fifth player, whose service rate 0.031 is greater than the arrival rate 0.028, still buys more bandwidth, since the bandwidth price at this time is very low (as known from the bandwidth cost function). As soon as the bandwidth becomes more expensive, the fifth player gives up and starts selling bandwidth, as shown in the curve which decreases for the first agent. All the other agents behave similarly: they buy till they have excess bandwidth, and then, when the cost per bandwidth is too high, they start to sell, till they reach their equilibrium. Note that we did not report in the figure the absolute time scale, that strongly depends on the absolute values of system parameters, and that is not significant in this example.

Let us now consider more quantitative simulation results. All the simulations ran for a fixed number of one million time slots, equivalent to 50 ms and to about 6-32 millions of switched cells, depending on the load. The initial transient period was estimated and not considered when collecting performance indices. The filter reactivity α, when needed in the cost functions, is set to 0.99, such that about 100 frames are required to "learn" the traffic arrival rates, equivalent to 0.6 ms and 5 ms for $F = 128$ and $F = 1024$, respectively.

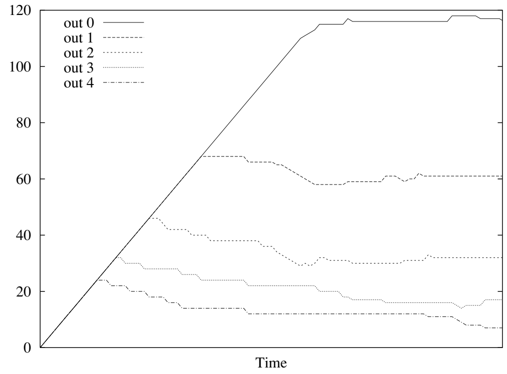

Fig. 1. Time evolution of the number of purchased slots by agents of input 0.

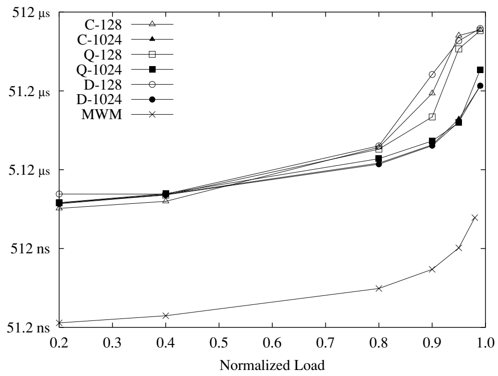

Fig. 2. Average delay for stationary uniform traffic. All SSB versions achieve the maximum throughput.

We considered three versions of the SSB policy, characterized by the use of different performance cost functions. We use the identifier C for "average queue length", D for "average delay" and Q for "queue length", according to the definitions of Sect. IV.

All the considered versions of SSB achieve 100% throughput in all traffic scenarios. We simulated also iSLIP [4] (with 32 iterations!) and observed that it achieves 83.1% maximum throughput under log-diagonal traffic and 96.8% under lin-diagonal traffic.

Fig. 2 and 3 show the average delay for uniform and lin-diagonal traffic respectively. The labels in the figures are composed by an identifier of the performance cost, which can be C (average queue length), D (average delay) and Q (queue length), followed by the frame size. When the frame size increases, at higher loads the delays decrease since the better granularity permit to better tune the number of services per frame. On the other hand, at lower loads, the delays increase, since the system state is updated only at frame boundaries. With respect to the throughput-optimal MWM slot

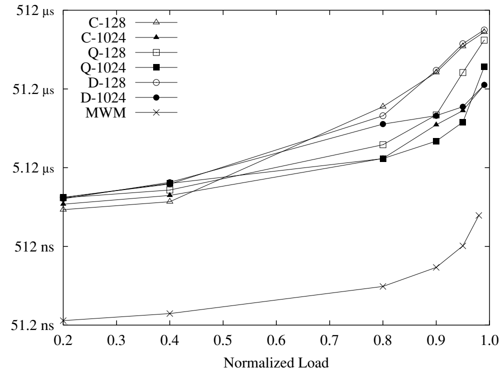

Fig. 3. Average delay for stationary lin-diagonal traffic. All SSB versions achieve the maximum throughput.

TABLE I
AVERAGE DELAY FOR TWO PERFORMANCE COST FUNCTIONS

Performance cost function	Time-variant traffic uniform/lin-diagonal	Stationary traffic uniform	lin-diagonal
Average delay	27.1 μs	12.0 μs	10.9 μs
Queue length	12.2 μs	12.5 μs	11.7 μs

by slot scheduler, the performance penalty is evident, although acceptable in absolute values when considering the real-time needs of delay sensitive applications.

B. Performance under time-variable traffic

We wish to show that, even if frame scheduling intrinsically implies slower time dynamics than slot-by-slot scheduling, still, its reactivity can be considered sufficient to track realistic changes in time-variant traffic.

We ran fixed simulation runs of about 82 millions of time slots with load 0.9 (corresponding to 4.2 seconds and about 2.5 billions of cells). The frame size is set to 1024, which corresponds to the slowest reactivity. The traffic is non-stationary, and cycles between two traffic matrices, $M^{(1)}$ and $M^{(2)}$ with period T. During a period of length T, four phases can be identified: in the first phase, the arrival rates are kept fixed according to $M^{(1)}$; in the second phase rates change linearly from $M^{(1)}$ to $M^{(2)}$; in the third phase, the traffic matrix is kept fixed and equal to $M^{(2)}$; finally, in the fourth phase rates change linearly from $M^{(2)}$ to $M^{(1)}$. More formally, the arrival rates $\lambda_{ij}(t)$ are given, for each period of length T, by:

$$\lambda_{ij}(t) = \begin{cases} M^{(1)}_{ij} \qquad \text{for } 0 \leq t < T/4 \\ M^{(1)}_{ij} + (t - T/4)/(T/4)(M^{(2)}_{ij} - M^{(1)}_{ij}) \\ \qquad\qquad \text{for } T/4 \leq t < T/2 \\ M^{(2)}_{ij} \qquad \text{for } T/2 \leq t < 3T/4 \\ M^{(2)}_{ij} + (t - 3T/4)/(T/4)(M^{(1)}_{ij} - M^{(2)}_{ij}) \\ \qquad\qquad \text{for } 3T/4 \leq t < T \end{cases}$$

We set $M^{(1)}$ to the uniform traffic matrix, $M^{(2)}$ to the lin-diagonal, and the period $T = 42$ ms, a value that was considered reasonable to detect realistic traffic dynamics even at the TCP level. For the average delay performance cost, we set $\alpha = 0.9$ in (3), equivalent to about 10 frames (1 ms) required to correctly estimate traffic changes.

No throughput losses were observed. In Table I, we show the average delays for non-stationary traffic under different performance costs. We report, in the right hand side of the table, the delays obtained in the case of stationary traffic, under uniform and lin-diagonal traffic only. It can be seen that the average delays are very satisfactory when using the queue length, whereas a small delay increase is experienced when using the average delay as performance cost. Thus, the SSB policy appears to be robust also when considering time-variant traffic.

C. Support of different priorities

Finally, we want to highlight the SSB policy flexibility: in the same framework, by simply weighting the cost factor of different agents, we can differentiate the experienced delays.

We ran our simulations for 52 ms, with $F = 1024$ cells, corresponding to about 6.4 million switched cells for $\rho = 0.2$, and to 28 million cells for $\rho = 0.9$.

In the first scenario, we define 2 priorities by assigning two values to a weight factor w_{ij} used as a multiplier of the performance cost function: $w_{ij} = 1$ for $i = 0, \ldots, N-1$, $d = 0, \ldots, N/2-1$ and $j = |i+d|_N$, where d is the diagonal identifier. Otherwise, $w_{ij} = 10$. That is, the first $N/2$ diagonals of the traffic matrix correspond to a cost factor which is 10 times lower than the other $N/2$ diagonals. Traffic of both priorities exists in all input and all output ports.

In the second scenario, we define N priorities: $w_{ij} = \alpha^d$, with $i = 0, \ldots, N-1$, $d = 0, \ldots, N-1$ and $j = |i+d|_N$. That is, each diagonal has a cost factor which is α times the cost factor of the next right diagonal. We set $\alpha = 0.8619$ to have a ratio of 100 between cost factors corresponding to the highest and the lowest priority diagonals. All input and output ports are loaded with the same amount of traffic

Performance results are shown in Fig. 4 for 2 priorities, and Fig. 5 for the N priorities. A very good control of average delays is obtained by using the weight coefficients.

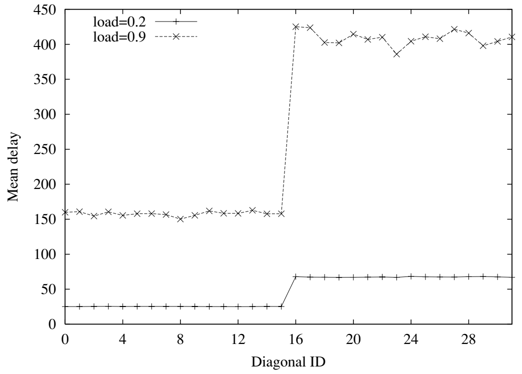

Fig. 4. Average delay for a system with 2 priority levels, as a function of the diagonal identifier

VII. MAIN FEATURES OF THE POLICY

We previously provided a number of arguments supporting differential frame-based approaches. We now want to highlight

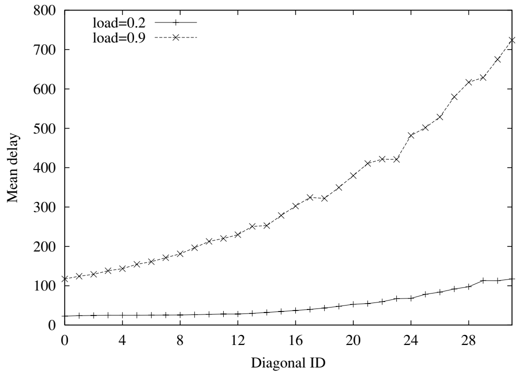

Fig. 5. Average delay for a system with N priority levels, as a function of the diagonal identifier

the reasons why we believe that the described framework constitutes a significant advantage with respect to other proposed approach relying on differential frame-based matching.

First, the policy is demonstrated to be stable, both for switches in isolation, and for networks of switches. By contrast, even the optimal slot-by-slot MWM algorithm is not stable in networks of switches. However, it is fairly easy to devise other, likely simpler, policies that are stable; for example, simply trying to maintain $x_{ij} = \hat{\lambda}_{ij}$ would be enough to obtain stability.

A second feature of our proposal is that it works robustly also with a rough estimation of traffic rates, obtained simply by looking at queue lengths.

Moreover, the proposed policy shows its most important advantage when looking at its flexibility. For example, by properly weighting the cost functions, a control on the relative delay performance among flows was shown to be easy to obtain. Several other features related to differentiated QoS needs may be introduced by properly weighting cost functions; from a practical point of view, this would require only to set some system parameters in order to obtain a different behavior without changing the overall scheduling scheme.

VIII. Conclusions

We proposed a new frame-based scheduling framework to solve the contention problem in accessing switching fabrics in high-speed IQ switches. The scheduling is defined in terms of a game among queues: a unique NEP is shown to exist. The NEP is efficient and the algorithm converges to the NEP when using a well defined set of cost functions. Our scheduling policy achieves 100% throughput under a large class of input traffic patterns; the same result holds also in a network of switches. In terms of performance, we showed by simulation that no throughput limitation exists, that good performance in terms of delays is obtained, and that the proposed framework is very flexible and permits to obtain differentiated delays for different flows.

References

[1] P.Pappu, J.Parwatikar, J.Turner, K.Wong, "Distributed Queueing in Scalable High Performance Routers", *IEEE INFOCOM'03*, San Francisco, CA, Apr.2003

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

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

[4] N.McKeown,"The iSLIP scheduling algorithm for input-queued switches", *IEEE/ACM Transactions on Networking*, vol.7, n.2, Aug.1999, pp.188-201

[5] D.N.Serpanos, P.I.Antoniadis, "FIRM: a class of distributed scheduling algorithms for high-speed ATM switches with multiple input queues", *IEEE INFOCOM'00*, Tel Aviv, Israel, Apr.2000, pp.548-555

[6] P.Giaccone, B.Prabhakar, D. Shah, "Towards simple, high-performance schedulers for high-aggregate bandwidth switches", *IEEE INFOCOM'02*, New York, NY, Jun.2002, pp.1160-1169

[7] J.Chao, "Saturn: a terabit packet switch using dual round robin", *IEEE Communications Magazine*, vol.38, n.12, Dec.2000, pp.78-84

[8] L.Tassiulas, "Linear complexity algorithms for maximum throughput in radio networks and input queued switches", *IEEE INFOCOM'98*, San Francisco, CA, Apr.1998, pp.553-559

[9] F.M.Chiussi, A.Francini, G.Galante, E.Leonardi, "A Novel Highly-Scalable Matching Policy for Input-Queued Switches with Multiclass Traffic", *IEEE GLOBECOM 2002*, Taipei, Taiwan, Nov. 2002

[10] A.Bianco, M.Franceschinis, S.Ghisolfi, A.Hill, E.Leonardi, F.Neri, R.Webb, "Frame-based matching algorithms for input-queued switches", *High Performance Switching and Routing (HPSR 2002)*, Kobe, Japan, May 2002

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

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

[13] M.Ajmone Marsan, P.Giaccone, E.Leonardi, F.Neri, "On the stability of local scheduling policies in networks of packet switches with input queues", *IEEE JSAC*, vol. 21, n. 4, pp.642-655, May 2003

[14] M.J.Osborne, A.Rubinstein, *A course in game theory*, MIT Press, London, 1994

[15] A.Varma, S.Chalasani, "An incremental algorithm for TDM switching assignments in satellite and terrestrial networks", *IEEE JSAC*, vol.10, n.2, Feb.1992, pp.364-377

[16] J.Y.N.Hui, *Switching and Traffic Theory for Integrated Broadband Networks*, Kluwer Academic Publishers, Jan.1990

[17] G.Debreu, *Theory of Value: An axiomatic analysis of economic equilibrium*, Wiley, New York, 1959