Cell Switching Versus Packet Switching in Input-Queued Switches

Yashar Ganjali, *Student Member, IEEE*, Abtin Keshavarzian, *Student Member, IEEE*, and Devavrat Shah

***Abstract*—Input Queued (IQ) switches have been well studied in the past two decades by researchers. The main problem concerning IQ switches is scheduling the switching fabric in order to transfer packets from input ports to output ports. Scheduling is relatively easier when all packets are of the same size. However, in practice, packets are of variable length. In the current implementation of switches, variable length packets are segmented into fixed length packets—also knowns as cells—for the purpose of scheduling. However, such cell-based switching comes with some significant disadvantages: (a) loss of bandwidth due to the existence of incomplete cells; and (b) additional overhead of segmentation of packets and re-assembly of cells. This is a strong motivation to study packet-based scheduling, i.e., scheduling the transfer of packets without segmenting them.**

The problem of packet scheduling was first considered by Marsan *et al.* They showed that under any admissible Bernoulli IID (independent and identically distributed) arrival traffic, a simple modification of the Maximum Weight Matching (MWM) algorithm achieves 100% throughput. In this paper, we first show that no work-conserving (i.e., maximal) packet-based algorithm is stable for arbitrary admissible arrival processes. Thus, the results of Marsan *et al.* are strongly dependent on the arrival distribution. Next, we propose a new class of "waiting" algorithms. We show that the "waiting"-MWM algorithm is stable for any admissible traffic using the fluid limit technique.

We would like to note that the algorithms presented in this paper are distribution independent or universal. The algorithms and proof methods of this paper may be useful in the context of other scheduling problems.

***Index Terms*—Cell switching, packet switching, scheduling, variable length packets.**

I. Introduction

TWO important design criteria for switching architectures are: (a) throughput of the system, and (b) average delay of packets. Among different switching architectures, the Input Queued (IQ) switch architecture has been very attractive due to its low memory bandwidth requirements compared to other known architectures. The crossbar constraints of an IQ switch require it to schedule packets to be transferred between inputs and outputs. The scheduling algorithm used to transfer packets affects the throughput and delay in an IQ switch. A tremendous amount of research has been done on the design of scheduling algorithms for IQ switches [1]–[3], [8]. All these studies make an implicit assumption that the switch works with fixed-size cells. In other words, they all assume that whenever a packet arrives to the system, it is divided into equal-sized cells, and after the switching is done, the cells are re-assembled in the form of the original packet before leaving the system. Contrary to this common assumption, we consider systems in which the switch directly works on packets without breaking them into cells. We call such a switching system a packet-based system, compared to traditional cell-based systems. Using fixed-size cells in the switch makes the implementation of the scheduling algorithm easier compared to the variable-length packets. However, cell-based switching has the following major disadvantages:

i) Packets arriving at input side need to be segmented into cells, requiring a special input segmentation module. Also, at the output side cells need to be re-assembled. This induces significant implementation overhead.

ii) Segmentation of packets can result in generation of some incomplete cells since a cell cannot contain data belonging to two different packets. This can cause a significant bandwidth loss. For instance, if the cell size is 64 bytes and packet size is 40 bytes then the amount of bandwidth loss is $24/64 \approx 37\%$.

Packet-based scheduling algorithms have been studied before [7]. It is important to first understand the stability region for the case of packet-scheduling algorithms. Naturally, there is some similarity between packet-based and cell-based scheduling. Cell-based scheduling is known to keep the system stable under the Maximum Weight Matching (MWM) algorithm [1]–[6] for any admissible traffic. In [7] it has been shown that canonical modification of the cell-based MWM algorithm into a packet-based algorithm, which we denote as PB-MWM, achieves 100% throughput for any admissible Bernoulli IID traffic with IID packet lengths having finite mean and variance.

In this paper, we first show a straightforward result that the PB-MWM is rate stable (see Section II) for any IID admissible traffic with IID packet lengths having finite mean and variance. We use the fluid model technique to prove the rate stability [6]. Surprisingly, when packet lengths are not IID, we find that any work-conserving packet-based scheduling algorithm, in particular PB-MWM, is not always stable. This observation suggests that there is a fundamental difference between packet-based and cell-based scheduling algorithms. Hence, we need to design a different type of packet-based scheduling algorithm in order to obtain stability for an arbitrary class of arrival processes.

Manuscript received May 1, 2003; revised September 8, 2004; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor J. Roberts.

Y. Ganjali is with the Department of Electrical Engineering, Stanford University, Stanford, CA 94305-9045 USA (e-mail: ganjali@ stanford.edu).

A. Keshavarzian was with the Department of Electrical Engineering, Stanford University, Stanford, CA 94305-9045 USA. He is now with the Research and Technology Center (RTC), Robert Bosch Corporation, Palo Alto, CA 94304 USA.

D. Shah was with the Department of Electrical Engineering, Stanford University, Stanford, CA 94305-9045 USA. He is now with the Department of Electrical Engineering and Computer Science and Engineering Systems, Massachusetts Institute of Technology, Cambridge, MA 02139 USA (e-mail: devarat@mit.edu).

Digital Object Identifier 10.1109/TNET.2005.852884

We propose a new class of scheduling algorithms which we call "waiting" algorithms. In particular, we show that the "waiting" modification of PB-MWM is stable for any admissible traffic such that the distribution of packet lengths is stationary and ergodic with finite mean and variance. We would like to note that the class of "waiting" algorithms is universal in the sense that they do not require knowledge of the packet size distribution nor do they try to *learn* the distribution.

In summary, the main contributions of this paper are as follows.

1) Work-conserving packet-based scheduling algorithms are not always stable.
2) Waiting algorithms are stable.

The structure of this paper is as follows. In Section II, we briefly describe the input-queued switch architecture, compare the packet-based and cell-based scheduling algorithms, and briefly describe the cell-based maximum weight matching (MWM) algorithm. In Section III we introduce the fluid model for the switch, and the dynamics of the switch. In Section IV, we obtain a general condition for rate stability (see Lemma 1). In Section V, we present a counter-example that motivates us to study a new type of packet-based algorithm, denoted as "waiting" algorithms. In Section VI, we introduce a simple waiting algorithm, which is proved to be stable. Finally, we conclude the paper in Section VII.

II. Input-Queued Switch

In this section we describe the Input-Queued (IQ) switch architecture along with useful definitions and notations.

Model of an IQ Switch: Fig. 1 shows the logical structure of an IQ switch. We assume that the switch has the same number of input and output ports,[1] denoted by N. Time is slotted. We assume that all the incoming and outgoing line-rates are the same. Let data that can arrive/depart in a time slot be denoted as "cell". Thus, in a given time slot at most one cell can arrive at and/or depart any port. A packet may consist of multiple cells and hence the first cell and the last cell belonging to a packet may arrive at different time slots.[2]

In order to eliminate the well-known head-of-line blocking problem, each input maintains N separate FIFO buffers one for each of the N outputs, denoted by "Virtual Output Queues" (VOQ). Data arriving at input i destined for output j is stored in VOQ_{ij}.

Switching Constraints and the Scheduling Problem: A switching unit, which is a cross-bar fabric, is required to transfer data from input ports to appropriate output ports. We say that a switch has speed up S, if at each time slot at most S cells can be removed from each input and at most S cells can be transferred to each output. The cross-bar fabric imposes the following scheduling constraints: In a given time slot

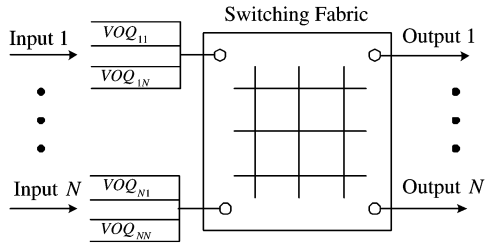

Fig. 1. An input-queued switch.

i) each input port can be connected to at most one output port; and
ii) each output port can be connected to at most one input port.

The "scheduler" determines which inputs and outputs are connected during each time slot. The scheduling problem can be viewed as a bipartite graph matching problem. A matching is represented by an $N \times N$ matrix $\mathbf{m} = [m_{ij}]$ where $m_{ij} = 1$ if input i is connected to output j, and $m_{ij} = 0$ otherwise. The set of all possible matchings is denoted by $\mathcal{M}$. A scheduling algorithm is "*work-conserving*" or "*maximal*", if an input is never left unmatched when it has data for an unmatched output.

Arrival Process: Let $A_{ij}(n)$ denote the number of cells that have arrived at input i destined to output j up to time n. We adopt the convention that $A_{ij}(0) = 0$. We assume that the arrival processes $\mathbf{A}(n) = [A_{ij}(n)]$ satisfy the strong law of large numbers (SLLN), i.e., for any $i, j = 1, \ldots, N$, almost surely,

$$\lim_{n\to\infty} \frac{A_{ij}(n)}{n} = \lambda_{ij}. \tag{1}$$

We call λ_{ij} the arrival rate at VOQ_{ij}. Note that this is a very mild restriction on the arrival process. The process $A(\cdot) = [A_{ij}(\cdot)]$ summarizes only net data arrived at ports. To completely describe the arrival process we additionally need: (i) Packet length distribution, (ii) Dependence between arrival times of packets over time, and (iii) Dependence among arrival times across the input ports.

In case of Bernoulli IID traffic, it is assumed that the packet lengths are IID; the arrival time of a new packet, after the completion of the previous packet, is independent and identically distributed; and the traffic is independent across inputs. A less realistic modification of this traffic was considered in [7], where a packet was assumed to arrive in a time slot containing multiple cells.

Next we define an admissible arrival process as follows.

Definition 1: The arrival process with arrival rate matrix $\mathbf{\Lambda} = [\lambda_{ij}]$ is called "*admissible*" iff (1) holds and no input or output is overloaded, in other words,

$$\sum_{i=1}^{N} \lambda_{ij} \le 1 \quad \forall\, j = 1, \ldots, N \tag{2}$$

$$\sum_{j=1}^{N} \lambda_{ij} \le 1 \quad \forall\, i = 1, \ldots, N. \tag{3}$$

If all the inequalities are strict, then the arrival rate matrix is called *strictly admissible*.

[1] In practice, one input and one output interface reside on the same line card, thus the numbers of input and output ports are the same.

[2] Similar to the assumption made in [7], we assume that a packet contains an integer number of cells. We would like to note that the results of [7] are crucially dependent on this assumption; our results for waiting algorithms (Theorems 2 and 3) can be easily extended for packet-sizes which are nonintegral.

Switch Throughput: Let $D_{ij}(n)$ be the number of departures from VOQ_{ij} up to time n. Again, let $D_{ij}(0) = 0$ and $\mathbf{D}(n) = [D_{ij}(n)]$.

Definition 2: A switch operating under a matching algorithm is said to have 100% throughput or is called *rate stable* if for any admissible arrival process $\mathbf{A}(n) = [A_{ij}(n)]$ with rate λ_{ij}, almost surely,

$$\lim_{n\to\infty} \frac{D_{ij}}{n} = \lambda_{ij} \quad \forall\, i,j = 1,\ldots,N. \tag{4}$$

Scheduling Algorithms: Let $Z_{ij}(n)$ be the number of cells in VOQ_{ij} at time n, including any arrival at time n, and $\mathbf{Z}(n) = [Z_{ij}(n)]$.

Definition 3: For any matching $\mathbf{m} \in \mathcal{M}$ the "*weight*" $W_{\mathbf{m}}(n)$ of the matching at time n is defined as

$$W_{\mathbf{m}}(n) = \langle \mathbf{m}, \mathbf{Z}(n) \rangle \tag{5}$$

where $\langle \mathbf{A}, \mathbf{B} \rangle = \sum_{ij} A_{ij}B_{ij}$ for two matrices $\mathbf{A}$ and $\mathbf{B}$ of the same size.

Intuitively, the weight of a matching is the sum of the sizes of the queues that are being served by this matching. Next, we define two different types of scheduling algorithms: cell-switching and packet-switching.

Definition 4: A packet-based scheduling algorithm is a scheduling algorithm such that once it starts transmitting the first cell of a packet to an output port, it continues the transmission until the whole packet is completely received at the corresponding output port. In other words, an input-output pair remains connected till all the cells corresponding to a single packet are transmitted.

Definition 5: A cell-based scheduling algorithm, in contrast to a packet-based scheduling algorithm, can change the matching even though some of the cells belonging to a packet are already transmitted to the corresponding output port while some are still at the input port.

We note that a packet-based scheduling algorithm avoids the problem of segmentation at input ports and reassembly of cells at output ports in a switch. In any cell-based switching system, different cells of the same packet may observe different delay values before leaving the system. It is reasonable to assume that the delay seen by the user is the same as the delay observed by the last cell of any packet. Therefore, a scheduling algorithm that transfers the last cell of a packet with larger delay performs poorly even if it performs well on all other cells of the packet. Most of the known cell-based scheduling algorithms are not aware of the existence of packets, and therefore there is a chance that a packet-based scheduling algorithm which is aware of the entity of a packet can use this information to do a better scheduling (in the sense of the waiting delay observed by the users). Authors in [7] gave similar reasoning in favor of packet scheduling.

Let us consider a natural way to convert a known cell-based algorithm into a packet-based algorithm. Let $\mathcal{X}$ be any cell-based scheduling algorithm, e.g., MWM, maximal matching. The packet-based version of $\mathcal{X}$, denoted by PB-$\mathcal{X}$ is as follows: **PB-$\mathcal{X}$**:

1) Let $\mathbf{m}(n)$ denote the schedule used at time n.
2) At time $n+1$
   a) if all ports are free then use $\mathbf{m}(n+1) = \boldsymbol{\mathcal{X}}(n+1)$.
   b) else set $\mathbf{m}(n+1) = \mathbf{m}(n)$.

Next, we describe an important cell-based scheduling algorithm, known as the Maximum Weight Matching (MWM) algorithm. In this paper, we will be interested in PB-MWM and its variants.

Maximum Weight Matching (MWM) Algorithm At each time slot, the MWM algorithm selects the matching with the maximum weight among all matchings in $\mathcal{M}$. If there are multiple such matchings, one of them is selected arbitrarily. Clearly, the MWM algorithm is a cell-based scheduling algorithm in this setting. We denote the maximum weight matching and its corresponding weight at time n by $\mathbf{m}^\star(n)$ and $W^\star(n)$ respectively, i.e.,

$$\mathbf{m}^\star(n) = \arg\max_{\mathbf{m}\in\mathcal{M}} W_{\mathbf{m}}(n) \tag{6}$$

$$W^\star(n) = \max_{\mathbf{m}\in\mathcal{M}} W_{\mathbf{m}}(n) = W_{\mathbf{m}^\star}(n). \tag{7}$$

In [1]–[3], it was shown that under any admissible Bernoulli IID traffic, MWM algorithm is stable. This result was extended by Dai-Prabhakar [6] for a larger class of admissible arrival traffic using the fluid model technique (for a weaker notion of stability). In this paper, we use fluid model techniques and adopt the notion of stability similar to [6] for the packet-based scheduling algorithm.

III. Fluid Model and Switch Dynamics

This section briefly describes the fluid model of a discrete time switch, which was first introduced in [6]. In [6] the following strong connection between the fluid model of a switch and the actual discrete time switch dynamics was established: (Theorem 3, [6]) The stability of the fluid model of the switch implies the stability of the discrete time switch. An interested reader can refer to [6] for details of the fluid model. In this paper, we will prove the stability of the fluid model under algorithms of interest and use the above result to infer the stability of the original discrete time switch.

For any $\mathbf{m} \in \mathcal{M}$, let $T_{\mathbf{m}}(n)$ represent the cumulative amount of time that the matching $\mathbf{m}$ has been used up to time n under the scheduling algorithm used. We assume that $T_{\mathbf{m}}(0) = 0$. Note that $T_{\mathbf{m}}(n)$ is a nondecreasing function with respect to n which depends on both the arrival process and the scheduling algorithm. The following three equations govern the dynamics of the switch:

$$Z_{ij}(n) = A_{ij}(n) - D_{ij}(n), \quad \forall\, i,j \tag{8}$$

$$D_{ij}(n) = \sum_{\mathbf{m}\in\mathcal{M}} m_{ij} 1_{(Z_{ij}(n)>0)} (T_{\mathbf{m}}(n) - T_{\mathbf{m}}(n-1)) + D_{ij}(n-1) \quad \forall\, i,j \tag{9}$$

$$\sum_{\mathbf{m}\in\mathcal{M}} T_{\mathbf{m}}(n) = n. \tag{10}$$

The first equation simply states that the number of cells in VOQ_{ij} at any time equals the total number of arrivals minus the total number of departures. The second equation states that the number of departures till time n equals the number departures

till previous time slot (time slot $n-1$) plus the departures at time n. Note that $T_{\mathbf{m}}(n)-T_{\mathbf{m}}(n-1)$ is either 1 if the matching $\mathbf{m}$ is used at time n, or 0 if it is not. Hence, $D_{ij}(n)$ increases if the matching used at time n connects input i to output j and the queue VOQ_{ij} is not empty. The third equation simply states that at each time slot, exactly one of the possible matchings is used.

From [6], the continuous equations governing the dynamics of the corresponding fluid model of the switch described above are as follows:

$$\tilde{Z}_{ij}(t) = \lambda_{ij}t - \tilde{D}_{ij}(t) \quad \forall\, i,j \tag{11}$$

$$\frac{\partial \tilde{D}_{ij}(t)}{\partial t} = \sum_{\mathbf{m}\in\mathcal{M}} m_{ij}\frac{\partial \tilde{T}_{\mathbf{m}}(t)}{\partial t} \quad \text{if } \tilde{Z}_{ij}(t)>0 \quad \forall\, i,j \tag{12}$$

$$\sum_{\mathbf{m}\in\mathcal{M}} \tilde{T}_{\mathbf{m}}(t) = t \tag{13}$$

where the functions $\tilde{Z}_{ij}(t)$, $\tilde{D}_{ij}(t)$, and $\tilde{T}_{\mathbf{m}}(t)$ are called the fluid limits and are obtained from the discrete random processes $Z_{ij}(n)$, $D_{ij}(n)$, and $T_{\mathbf{m}}(n)$. For example, $\tilde{Z}_{ij}(t)$ is obtained as follows. First, we create $\hat{Z}_{ij}(t)$ which is a continuous version of the discrete function $Z_{ij}(n)$

$$\hat{Z}_{ij}(t) = Z_{ij}(\lfloor t\rfloor) + [Z_{ij}(\lfloor t+1\rfloor) - Z_{ij}(\lfloor t\rfloor)](t-\lfloor t\rfloor). \tag{14}$$

Then the fluid limit is obtained as follows:

$$\tilde{Z}_{ij}(t) = \lim_{r\to\infty}\frac{\hat{Z}_{ij}(rt)}{r}. \tag{15}$$

All other fluid limit functions are obtained in a similar manner, i.e., the time is scaled by r and the function is re-normalized by dividing by r and we let $r\to\infty$. Thus, the fluid model of the switch is described by vector $(\tilde{\mathbf{Z}}(\cdot),\tilde{\mathbf{D}}(\cdot),\tilde{\mathbf{T}}(\cdot))$, where $\tilde{\mathbf{Z}}(\cdot)=[\tilde{Z}_{ij}(\cdot)]$, $\tilde{\mathbf{D}}(\cdot)=[\tilde{D}_{ij}(\cdot)]$, $\tilde{\mathbf{T}}(\cdot)=[\tilde{T}_{\mathbf{m}}(\cdot)]$.

The following is the main result of [6] that we will use in this paper.

Theorem 1 (Theorem 3, [6]): A switch operating under an algorithm is rate-stable if for the corresponding fluid model $\tilde{Z}(t)=0$ for almost all t given $\tilde{Z}(0)=0$.

IV. General Stability Condition

In this section, we derive a general condition which implies rate stability of the switch scheduling algorithm. We will first use this condition to obtain a different proof of rate-stability of the PB-MWM algorithm. Then, in the following sections we will use the condition to obtain stability of other packet-based scheduling algorithms. The condition is stated as follows.

Lemma 1: Consider a switch operating under a scheduling algorithm $\mathcal{A}$. Let $\mathbf{m}(n)$ denote the schedule at time n with $W(n)$ as its weight. Let $W^{\star}(n)$ denote the weight of maximum weight schedule at time n, with the same weights (queue-sizes). Further, let

$$\mathbb{E}\{|W(n)-W^{\star}(n)|\}\le B \tag{16}$$

where expectation $\mathbb{E}\{\cdot\}$ is taken over the probability space containing information about the arrival process and randomness in $\mathcal{A}$. Then, the algorithm $\mathcal{A}$ is rate-stable.

Proof: We prove this Lemma using Theorem 1. As described above, the fluid model of the switch is described by the vector $(\tilde{\mathbf{Z}}(\cdot),\tilde{\mathbf{D}}(\cdot),\tilde{\mathbf{T}}(\cdot))$. Under algorithm $\mathcal{A}$ we will show that $\tilde{\mathbf{Z}}(t)=[0]$ for almost all t, given $\tilde{\mathbf{Z}}(0)=[0]$.

Define a Lyapunov function $L(t)$ as

$$L(t) = \langle \tilde{\mathbf{Z}}(t),\tilde{\mathbf{Z}}(t)\rangle = \sum_{i,j}\tilde{Z}_{ij}^2(t). \tag{17}$$

We first note the following Lemma.

Lemma 2 (Lemma 1, [6]): Let $f\colon[0,\infty)\to[0,\infty)$ be an absolutely continuous function with $f(0)=0$. Assume that $\dot{f}(t)\le 0$ for almost all t (with respect to Lebesgue measure) such that $f(t)>0$ and f is differentiable at t. Then $f(t)=0$ for almost every $t\ge 0$.

In [6] the authors showed that for algorithm MWM, $\dot{L}(t)\le 0$ whenever $L(t)>0$ (equivalently, if any of $\tilde{Z}_{ij}>0$). Then, using Lemma 2 and Theorem 1, they concluded that MWM is rate-stable. Hence, in our case, we need to show that $\dot{L}(t)\le 0$ for almost all t whenever $L(t)\ge 0$ and $L(t)$ is differentiable at t, under algorithm $\mathcal{A}$ to prove the rate stability of $\mathcal{A}$.

Consider all t such that the fluid quantities are differentiable and hence $\dot{L}(t)$ is well defined. By definition,

$$\begin{aligned}\frac{\partial L(t)}{\partial t} &= 2\left\langle \frac{\partial\tilde{\mathbf{Z}}(t)}{\partial t},\tilde{\mathbf{Z}}(t)\right\rangle\\ &= 2\left\langle \mathbf{\Lambda}-\frac{\partial\tilde{\mathbf{D}}(t)}{\partial t},\tilde{\mathbf{Z}}(t)\right\rangle\\ &= 2\langle\mathbf{\Lambda},\tilde{\mathbf{Z}}(t)\rangle - 2\left\langle\tilde{\mathbf{Z}}(t),\frac{\partial\tilde{\mathbf{D}}(t)}{\partial t}\right\rangle.\end{aligned} \tag{18}$$

Substituting (12) we obtain

$$\begin{aligned}\left\langle\tilde{\mathbf{Z}}(t),\frac{\partial\tilde{\mathbf{D}}(t)}{\partial t}\right\rangle &= \sum_{\mathbf{m}\in\mathcal{M}}\langle\tilde{\mathbf{Z}}(t),\mathbf{m}\rangle\frac{\partial\tilde{T}_{\mathbf{m}}}{\partial t}\\ &= \sum_{\mathbf{m}\in\mathcal{M}}\tilde{W}_{\mathbf{m}}(t)\frac{\partial\tilde{T}_{\mathbf{m}}}{\partial t}\end{aligned} \tag{19}$$

where $\tilde{W}_{\mathbf{m}}(t)=\langle\tilde{\mathbf{Z}}(t),\mathbf{m}\rangle$.

Let us define $\Delta(n)$ as the difference between the weight of the MWM and the weight of the matching obtained by scheduling algorithm at time n. We know that $\mathbb{E}(\Delta(n))$ is bounded by some constant B which does not depend on n, and $\Delta(n)$ is a positive random variable. Hence, $\Delta(n)$ is bounded almost surely. Thus, on the fluid limit scale we obtain that

$$\tilde{\Delta}(t) = \lim_{r\to\infty}\frac{\hat{\Delta}(rt)}{r}\le\lim_{r\to\infty}\frac{\hat{B}}{r}=0. \tag{20}$$

Thus, in the fluid scale the weight of the MWM and the weight of the matching used by the scheduler will be the same, i.e.,

$$\tilde{W}_{\mathbf{m}}(t)=\tilde{W}^{\star}(t). \tag{21}$$

Therefore, the algorithm will only use the matchings that have the same weight as the maximum weight matching. If we denote

the set of matchings used by the scheduling algorithm by $\mathcal{M}'$, we get

$$\left\langle \tilde{\mathbf{Z}}(t), \frac{\partial \tilde{\mathbf{D}}}{\partial t} \right\rangle = \sum_{\mathbf{m}\in\mathcal{M}'} \tilde{W}^{\star}(t) \frac{\partial \tilde{T}_{\mathbf{m}}(t)}{\partial t} = \tilde{W}^{\star}(t) \sum_{\mathbf{m}\in\mathcal{M}'} \frac{\partial \tilde{T}_{\mathbf{m}}(t)}{\partial t}. \quad (22)$$

Note that although $\mathcal{M}' \subseteq \mathcal{M}$ since $\mathcal{M}'$ is the set of matchings used by the scheduler, we can modify (13) to

$$\sum_{\mathbf{m}\in\mathcal{M}'} \tilde{T}_{\mathbf{m}}(t) = t \quad (23)$$

changing $\mathbf{m} \in \mathcal{M}$ to $\mathbf{m} \in \mathcal{M}'$. Now combining this result with (22) we obtain

$$\left\langle \tilde{\mathbf{Z}}(t), \frac{\partial \tilde{\mathbf{D}}}{\partial t} \right\rangle = \tilde{W}^{\star}(t). \quad (24)$$

Hence, from (18) the derivative of $L(t)$ will be

$$\frac{\partial L(t)}{\partial t} = 2\langle \mathbf{\Lambda}, \tilde{\mathbf{Z}}(t) \rangle - 2\tilde{W}^{\star}(t). \quad (25)$$

From Birkoff–von Neumann's theorem we know that any doubly sub-stochastic (admissible) traffic matrix $\mathbf{\Lambda}$ can be majorized by a weighted sum of finite permutation (matching) matrices, i.e., we can find $\gamma_k > 0$ and $\mathbf{m}_k \in \mathcal{M}$ for $k = 1, \ldots, K$ such that

$$\mathbf{\Lambda} \preceq \sum_{k=1}^{K} \gamma_k \mathbf{m}_k, \quad \sum_{k=1}^{K} \gamma_k < 1 \quad (26)$$

where $\mathbf{A} \preceq \mathbf{B}$ iff $\forall\, i, j a_{ij} \leq b_{ij}$ ($\mathbf{A} = [a_{ij}]$ and $\mathbf{B} = [b_{ij}]$).

By definition of the maximum weight matching, we get

$$\langle \tilde{\mathbf{Z}}(t), \mathbf{m}_k \rangle \leq \tilde{W}^{\star}(t). \quad (27)$$

Combining (26), (25), (27), we obtain

$$\begin{aligned} \frac{\partial L(t)}{\partial t} &\leq 2\left\langle \tilde{\mathbf{Z}}(t), \sum_{k=1}^{K} \gamma_k \mathbf{m}_k \right\rangle - 2\tilde{W}^{\star}(t) \\ &= 2\sum_{k=1}^{K} \gamma_k \langle \tilde{\mathbf{Z}}(t), \mathbf{m}_k \rangle - 2\tilde{W}^{\star}(t) \\ &\leq 2\left(\sum_{k=1}^{K} \gamma_k - 1 \right) \tilde{W}^{\star}(t) \\ &\leq 0. \end{aligned} \quad (28)$$

Hence, if any $\tilde{Z}_{ij} > 0$ then $\tilde{W}^{\star}(t) \neq 0$ and therefore $\dot{L}(t) < 0$, and this completes the proof. ■

Next, we apply Lemma 1 to prove the rate stability of PB-MWM. Formally, we state the following theorem.

Theorem 2: Under any admissible Bernoulli IID arrival pattern (IID packet lengths of finite mean and variance) the PB-MWM algorithm is rate-stable.

Proof: To use Lemma 1, we need to show that under Bernoulli IID arrival, for the PB-MWM algorithm, for any time n

$$\mathbb{E}\{|W(n) - W^{\star}(n)|\} \leq B$$

where $W(n), W^{\star}(n)$ are weights of matching used by PB-MWM and the MWM matching respectively; and B is some finite constant. Due to stationarity of the arrival process and PB-MWM being a discrete deterministic function of arrival process, the $W(\cdot), W^{\star}(\cdot)$ are stationary. Hence, we consider a stationary time $n = 0$ with the switch assumed to have started at time $-\infty$. Let $\cdots < T_{-1} < T_0 \leq 0 < T_1 < T_2 < \cdots$ denote the random time instances when PB-MWM re-configures its matchings. For Bernoulli IID traffic, the random variables $S_i = T_i - T_{i-1}, i \in \mathbb{Z}$ are identically distributed. Further, Lemma 2 of [7] shows that under Bernoulli IID traffic (with finite mean and variance of IID packet lengths), $\mathbb{E}\{S_i^2\} < C$, where C is a constant that depends on the packet length distribution and N (size of the switch).

Under PB-MWM, the weight $W(0)$ is the weight of the matching that was MWM at time T_0. Since at most one cell arrives (departs) at each input (output) port (i.e., N cells arrive/depart in the whole switch) in a time slot, the following is easy to obtain:

$$\begin{aligned} W(0) &\geq W^{\star}(0) - 2(-T_0)N \\ &\geq W^{\star}(0) - 2(T_1 - T_0)N \end{aligned} \quad (29)$$

where $T_1 > 0$ by definition. Thus,

$$|W^{\star}(0) - W(0)| \leq 2S_1 N. \quad (30)$$

The above bound (30) is conditional on the event that 0 lies in the interval $(T_0, T_1]$. The stationary point 0, lies in this interval with probability proportional to S_1. Hence, the average time-stationary bound on the quantity of interest is given as

$$\begin{aligned} \mathbb{E}\{|W^{\star}(0) - W(0)|\} &\leq 2\mathbb{E}\left\{S_1^2\right\} N \\ &\leq 2CN. \end{aligned} \quad (31)$$

Inequality (31) satisfies the desired condition of Lemma 1, which in turn implies the statement of Theorem 2. ■

V. Instability of Work-Conserving Algorithms

In [6], the authors showed that the cell-based MWM algorithm is rate stable under any admissible arrival process as long as it satisfies the SLLN property as in (1). As shown in [7] (and Theorem 2), the PB-MWM is stable for Bernoulli IID arrival traffic. A natural question is whether the PB-MWM is stable for arbitrary admissible arrival processes. In this section, we show that any work-conserving packet-based scheduling algorithm is not always stable. This in turn implies that the PB-MWM algorithm is not necessarily stable.

Counter-Example for PB-MWM: Consider a switch operating under PB-MWM with input traffic pattern as shown in Fig. 2. The A_{ij} $(i, j = 1, 2)$ shows the arrivals to VOQ_{ij} over time. The traffic pattern is periodic with period equal to 10. Note that no input or output is overloaded. In fact $\lambda_{1,1} = 0.8$,

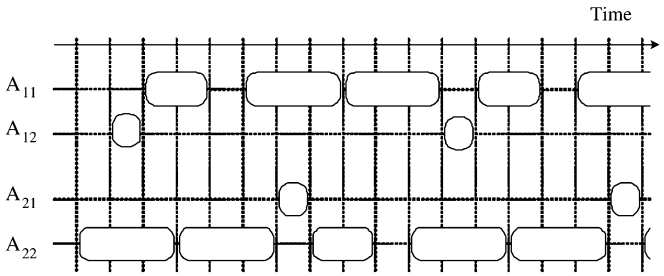

Fig. 2. The above figure depicts the exact traffic pattern under which PB-MWM algorithm is not stable.

$\lambda_{1,2} = 0.1$, $\lambda_{2,1} = 0.1$, and $\lambda_{2,2} = 0.8$. The switch can use one of two possible matchings, namely $\mathbf{m}_1$ which is called the parallel matching and $\mathbf{m}_2$ the cross matching, i.e.,

$$\mathbf{m}_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad \mathbf{m}_2 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}. \tag{32}$$

When the first packet arrives to the switch, the PB-MWM uses parallel matching $(\mathbf{m}_1)$, and then the scheduler is forced to keep the same matching for 3 time slots till the packet finishes. Before this packet is finished, a packet of length 2 comes to input 1 and it is scheduled for output 1 under the scheduling algorithm. In this way, under this traffic pattern, it is easy to see that whenever one input port is free, the other input port is busy serving a packet. In other words, the two ports are never free simultaneously. This forces the scheduling algorithm to use the parallel schedule all the time. Therefore, none of the packets arriving at VOQ_{12} and VOQ_{21} will ever get the chance to depart. Thus, the switch is unstable. Note that cell-based MWM will be able to handle this traffic.

Classification of Packet-Based Algorithms: The counter-example described above also shows that any work-conserving or maximal algorithm is not stable for that particular traffic pattern.

Theorem 1: There is no work-conserving packet-based scheduling algorithm that is rate stable for arbitrary admissible arrival processes.

The next question is if there is any packet-based algorithm that is stable? The above theorem suggests that to obtain a stable scheduling algorithm, if any, we need to consider nonwork conserving algorithms. Broadly speaking, the following are the two class of packet-based algorithms:

1) *Work-conserving (nonwaiting) algorithms*: under these algorithms an input is never left un-matched when it has a packet for any unmatched output port.
2) *Waiting algorithms*: these algorithms are not always work-conserving, i.e., they *wait* (do not start sending the packet although both input and output ports are free) for an *indefinite* number of time slots.

Note: The above result shows a fundamental difference between packet-based switches and cell-based switches. For cell-based switches being work-conserving is generally considered to be a good property. We know that MWM, which is generally stable, is work-conserving. Theorem 1 states that a generally stable packet-based scheduling algorithm should not be work-conserving. In other words, waiting has an essential role in the design of generally stable packet-based algorithms.

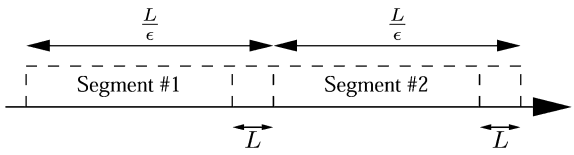

Fig. 3. Time segments in PB-wMWM. The algorithm simply waits in the second portion (of length L) of each interval.

VI. Stable Waiting Algorithm

In this section we describe MWM based waiting algorithms that achieve 100% throughput for any admissible traffic pattern. In particular, this algorithm is for the traffic pattern described in the previous section for which PB-MWM or any packet-based work-conserving scheduling policy was unstable.

The waiting algorithms are motivated by the counter-example described in the previous section for work-conserving algorithms. The main problem is that the work-conserving algorithm greedily matches the ports whenever possible, forcing it to always keep the parallel matching in the counter-example of Fig. 2. One way to overcome this problem is the following: when a packet gets served, do not schedule the freed ports till all ports become free and schedule according to a full MWM schedule. The waiting synchronizes the weight of the schedule to the weight of the MWM schedule. Hence, if *waiting is done frequently enough*, then, similar to the reasoning in the Proof of Theorem 2, we can verify that the weight of the schedule is always not more than a bounded constant away from MWM. However, we note that during the waiting period some ports lose bandwidth. Hence, if waiting is done too aggressively then the algorithm can not utilize the full bandwidth. These observations lead to the following waiting algorithm which we denote as PB-wMWM.

PB-wMWM:

- The switch runs at speedup $(1+\epsilon)$ for an arbitrarily small positive constant $\epsilon > 0$.
- Let the maximum length of any packet be bounded above by L.
- Divide the time into intervals of length $\frac{L}{\delta(\epsilon)}$ units, where $\delta(\epsilon) = (\frac{\epsilon}{1+\epsilon})$; i.e., time-intervals are $[0, \frac{L}{\delta(\epsilon)}]$, $[\frac{L}{\delta(\epsilon)} + 1, 2\frac{L}{\delta(\epsilon)}]$, and so on.
- At any time $n \in [k\frac{L}{\epsilon} + 1, (k+1)\frac{L}{\epsilon}]$ if all ports are busy serving previously scheduled packets, then do nothing. Else do the following:
  1) If $n \in [k\frac{L}{\delta(\epsilon)} + 1, (k+1)\frac{L}{\delta(\epsilon)} - L]$ use the usual PB-MWM to match the free ports as before.
  2) Else, wait till all packets currently scheduled to be serviced get served, which happens before the end of the current interval. Once all ports are free, use MWM to re-schedule all the ports.

Fig. 3 pictorially describes the induced time intervals by algorithm PB-wMWM.

The following Theorem proves the rate stability of PB-wMWM.

Theorem 2: The PB-wMWM algorithm is stable (rate stable) under any admissible traffic (with property (1)) with a known bound on maximum packet length at speedup $(1+\epsilon)$ for any $\epsilon > 0$.

Proof: The algorithm PB-wMWM is of waiting type, and therefore loses some bandwidth. This may result in instability. To prove the claimed result we show that the speedup compensates for the lost bandwidth and schedules of PB-wMWM satisfy the conditions of Lemma 1.

Loss of Bandwidth and Speedup: In PB-wMWM each port is maintained idle (not scheduled) by the algorithm for at most L time slots in every interval of length $L/\delta(\epsilon)$. That is, the fraction of the bandwidth lost is $\delta(\epsilon)$. Equivalently, each port is scheduled (may be empty) at least $1-\delta(\epsilon) = 1/(1+\epsilon)$ fraction of the time.

Under speedup $(1+\epsilon)$, if an algorithm schedules all ports all the time, (13) changes to

$$\sum_{\mathbf{m}\in\mathcal{M}} \tilde{T}_{\mathbf{m}}(t) = (1+\epsilon)t. \tag{33}$$

But since our algorithm is waiting, the above equation may not be true. But at worst, our algorithm schedules $1/(1+\epsilon)$ fraction of the time. Hence, (13) or (33) can be re-written as

$$\sum_{\mathbf{m}\in\mathcal{M}} \tilde{T}_{\mathbf{m}}(t) \geq t. \tag{34}$$

That is,

$$\sum_{\mathbf{m}\in\mathcal{M}} \frac{\partial \tilde{T}_{\mathbf{m}}(t)}{\partial t} \geq 1. \tag{35}$$

PB-wMWM Approximates MWM:

Note that every $L/\delta(\epsilon)$ time slots, the algorithm PB-wMWM re-configures itself to MWM schedule. Hence, using arguments similar to those in the Proof of Theorem 2, it is easy to see that for any time n

$$|W(n) - W^{\star}(n)| \leq 2LN/\delta(\epsilon) \tag{36}$$

where $W(n)$ and $W^{\star}(n)$ are the weights of PB-wMWM matching and MWM, respectively.

Finally, using (35), (36) and Lemma 1 implies that PB-wMWM is rate stable. This completes the Proof of Theorem 2. ∎

Note: As $\epsilon \to 0$, the PB-wMWM becomes PB-MWM. Thus, there is a "discontinuity" at $\epsilon = 0$.

A. Universal Stable Algorithm

The PB-wMWM assumes that the packet lengths are bounded and the bound is known to the algorithm. In reality this might not be the case. To address this issue, next we present a modification of the PB-wMWM algorithm, which is universal in the sense that it does not require any knowledge of the packet length distribution. We denote this algorithm by PB*-wMWM.

PB*-wMWM:

- The switch runs at speedup $(1+\epsilon)$ for an arbitrarily small positive constant $\epsilon > 0$.
- Initialization:
  1) Schedule the packets according to the MWM algorithm.
  2) If a packet on any port gets served, do not schedule a new packet on that port and wait until all ports finish their first set of scheduled packets.
  3) The maximum amount of idling done by any port is at most the maximum length of the packets that were scheduled initially, let it be $L_e(1)$.
  4) Set $M(1) = \frac{L_e(1)}{\delta(\epsilon)}$.
  5) Set $K = 1$.
- For all K, repeat the following:
  1) Run PB-MWM for next $M(K)$ time slots, and after $M(K)$ time slots, wait on any empty port till all ports are free.
  2) Let $L_e(K+1)$ be the maximum length of the packets under schedule when waiting starts at the end of $M(K)$ time slots.
  3) Set $M(K+1) = L_e(K+1)/\delta(\epsilon)$.
  4) Set $K = K+1$.

We can state the following Theorem about stability of PB*-wMWM.

Theorem 3: The PB*-wMWM algorithm is stable (rate stable) under any admissible traffic (with property (1)) at speedup $(1+\epsilon)$ for any $\epsilon > 0$ as long as the maximum packet length is finite (but unknown to the algorithm).[3]

Proof: We will use arguments very similar to the ones used in the Proof of Theorem 2.

The two main properties required in the Proof of Theorem 2 are: (a) The effective speed is at least 1; and (b) The weight of the schedule used by the algorithm is at most a bounded constant away from the weight of MWM. In the above algorithm, let us compute these two quantities as follows:

Loss of Bandwidth: The time is divided into intervals as follows: For $K \geq 2$, the Kth interval starts at the end of $M(K-1)$ time slots of the algorithm. It contains the time when some ports are idling subsequently and then $M(K)$ time slots. This is in contrast to the way intervals were defined in algorithm PB-wMWM, where idling came at the end of the interval rather than at the beginning. The time before the beginning of the second interval is called the first interval.

Since we are looking for the fraction of bandwidth loss asymptotically, we can neglect the first interval. For $K \geq 2$, at the Kth interval, each port is idle for at most $L_e(K)$ time slots while the algorithm schedules all ports for $L_e(K)/\delta(\epsilon)$ slots. Hence, using arguments similar to the ones used in the Proof of Theorem 2, we obtain that the fraction of bandwidth lost is at most $\epsilon/(1+\epsilon)$. Since the speedup is $(1+\epsilon)$, the effective throughput is at least 1.

PB*-wMWM Approximates MWM: Again, using arguments similar to the ones used in the Proof of Theorem 2, we obtain that for any time n in the Kth interval

$$\begin{aligned} |W(n) - W^{\star}(n)| &\leq 2L_e(K)N/\delta(\epsilon) \\ &\leq 2LN/\delta(\epsilon) \end{aligned} \tag{37}$$

[3]Theorem holds even for packet length with ergodic distribution of finite mean and variance.

where L is the bound on the maximum packet length, which is not known to the algorithm.

The above discussion and (37) imply the statement of the Theorem 3 by following the line of arguments as in the Proof of Theorem 2. ■

B. PB-MWM With Known $\mathbf{\Lambda}$

If the arrival rate matrix $\mathbf{\Lambda}$ is known and strictly admissible, then we can find a stable algorithm which runs with speedup one. Let σ be defined as follows:

$$\sigma = \max\left\{ \max_{i=1}^{N} \sum_{j=1}^{N} \lambda_{ij}, \max_{j=1}^{N} \sum_{j=1}^{N} \lambda_{ij} \right\}. \tag{38}$$

In other words, σ is the maximum arrival rate to any input port or leaving any output port. For any strictly admissible arrival traffic, σ is strictly less than 1. Then, on average, any input or output port is required to be busy for at most σ fraction of time slots in order to have stability. That is, on average an algorithm can idle any input/output port for $\epsilon < 1 - \sigma$ fraction of the time and still be stable. This leads to the following speedup 1 waiting algorithm, also denoted by PB-σwMWM, which is a straightforward modification of PB-wMWM. Similarly, we can obtain the speedup 1 algorithm, PB*-σwMWM, which is a modification of the universal algorithm PB*-wMWM.

PB-σwMWM:

- Let L be a bound on the maximum packet length and let $M > L$ be an integer such that $\frac{L}{M-L} < 1 - \sigma$, i.e., $M > \frac{(2-\sigma)L}{1-\sigma}$.
- Use the algorithm PB-wMWM with length of time interval as M at speedup 1.

The following is a straightforward corollary of Theorem 2.

Corollary 1: The PB-σwMWM algorithm is stable (rate stable) under any strictly admissible traffic (with property (1)) with known bound on maximum packet length and known σ, the maximum loading on any input/output port.

VII. Conclusion

In this paper we considered the packet-scheduling algorithms for IQ switch architectures. The results of [7] showed that modification of cell-based MWM for packet scheduling yields 100% throughput for any admissible Bernoulli IID traffic with independent packet lengths of bounded mean. We generalized this result for a somewhat broader class of arrival traffic pattern. We showed that there exists an admissible traffic pattern for which no work-conserving or maximal algorithm is stable. To overcome this problem we proposed a new class of waiting algorithms. Under the waiting algorithm the switch becomes stable for any admissible traffic. This was proved using the fluid limit technique. It is interesting to note that, unlike cell-based scheduling, work-conservation for packet scheduling is not always beneficial in this sense. This observation suggests that packet-based scheduling is quite different from cell-based scheduling.

Acknowledgment

The authors would like to thank B. Prabhakar and N. McKeown for useful discussions. They would also like to thank Dr. J. Roberts, H. Emrani, and the anonymous referees for their suggestions to improve the presentation of the paper.

References

[1] N. McKeown, V. Anantharam, and J. Walrand, "Achieving 100% throughput in an input-queued switch," in *Proc. IEEE INFOCOM*, 1996, pp. 296–302.

[2] L. Tassiulas and A. Ephremides, "Stability properties of constrained queuing systems and scheduling policies for maximum throughput in multi-hop radio networks," *IEEE Trans. Autom. Control*, vol. 37, no. 12, pp. 1936–1948, Dec. 1992.

[3] N. McKeown, V. Anantharam, and J. Walrand, "Achieving 100% throughput in an input-queued switch," *IEEE Trans. Commun.*, vol. 47, no. 8, pp. 1260–1267, Aug. 1999.

[4] N. McKeown, "iSLIP: A scheduling algorithm for input-queued switches," *IEEE Trans. Netw.*, vol. 7, no. 2, pp. 188–201, Apr. 1999.

[5] ——, "Scheduling Algorithms for Input-Queued Cell Switches," Ph.D. dissertation, Univ. of California, Berkeley, CA, May 1995.

[6] J. G. Dai and B. Prabhakar, "The throughput of data switches with and without speedup," in *Proc. IEEE INFOCOM*, 2000, pp. 556–564.

[7] M. A. Marsan, A. Bianco, P. Giaccone, E. Leonardi, and F. Neri, "Packet scheduling in input-queued cell-based switches," in *Proc. IEEE INFOCOM*, 2001, pp. 1085–1094.

[8] P. Giaccone, B. Prabhakar, and D. Shah, "Toward simple, high-performance schedulers for high-aggregate bandwidth switches," in *Proc. IEEE INFOCOM*, 2002, pp. 1160–1169.

Yashar Ganjali (S'03) received the B.Sc. degree in computer engineering from Sharif University of Technology, Tehran, Iran, in 1999, and the M.Sc. degree in computer science from the University of Waterloo, Waterloo, Canada, in 2001. Since then, he has been with the High Performance Networking Group at Stanford University, Stanford, CA, working toward the Ph.D. degree in electrical engineering.

His research interests include packet switching architectures/algorithms, wireless networking and optical networking.

Abtin Keshavarzian (S'99) was born in 1977. He received the B.S. and M.S. degrees (with honors) in electrical engineering from Sharif University of Technology, Tehran, Iran, in 1999 and 2001, respectively, and the Ph.D. degree in electrical engineering from Stanford University, Stanford, CA, in Winter 2004.

He is currently with the Research and Technology Center (RTC), Robert Bosch Corporation, Palo Alto, CA. His research interests include wireless sensor networks, packet switching algorithms and optical networks.

Devavrat Shah received the B.Tech. degree from the Indian Institute of Technology (IIT), Bombay, in 1999, and the Ph.D. degree in computer science from Stanford University, Stanford, CA, in Fall 2004.

He is currently an Assistant Professor in the Department of Electrical Engineering and Computer Science and the Engineering Systems Division at the Massachusetts Institute of Technology (MIT), Cambridge. His research interests are in the areas of network algorithms and large-scale networks such as wireless sensor networks.

Dr. Shah received the President of India Gold Medal at IIT-Bombay in 1999. He received the Best Paper Award at the IEEE INFOCOM 2004.