Input Queued Switches: Cell Switching vs. Packet Switching

Yashar Ganjali, Abtin Keshavarzian, Devavrat Shah
Department of EE and CS
Stanford University
Stanford, CA 94305
{abtink,yganjali,devavrat}@stanford.edu

***Abstract*— Input Queued(IQ) switches have been very well studied in the recent past. The main problem in the IQ switches concerns scheduling. The main focus of the research has been the fixed length packet-known as cells-case. The scheduling decision becomes relatively easier for cells compared to the variable length packet case as scheduling needs to be done at a regular interval of fixed cell time. In real traffic dividing the variable packets into cells at the input side of the switch and then re-assembling these cells into packets on the output side achieve it. The disadvantages of this cell-based approach are the following: (a) bandwidth is lost as division of a packet may generate incomplete cells, and (b) additional overhead of segmentation and reassembling cells into packets. This motivates the packet scheduling: scheduling is done in units of arriving packet sizes and in non-preemptive fashion. In [7] the problem of packet scheduling was first considered. They show that under any admissible Bernoulli i.i.d. arrival traffic a simple modification of Maximum Weight Matching (MWM) algorithm is stable, similar to cell-based MWM [1-4]. In this paper, we study the stability properties of packet based scheduling algorithm for general admissible arrival traffic pattern. We first show that the result of [7] extends to general re-generative traffic model instead of just admissible traffic, that is, packet based MWM is stable. Next we show that there exists an admissible traffic pattern under which any work-conserving (that is maximal type) scheduling algorithm will be unstable. This suggests that the packet based MWM will be unstable too. To overcome this difficulty we propose a new class of "waiting" algorithms. We show that "waiting"-MWM algorithm is stable for any admissible traffic using fluid limit technique [6].**

I. INTRODUCTION

The two important design criteria for switching architectures are: (a) throughput of the system, and (b) average delay. Among different switching architectures 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 requires it to schedule packets to be transferred between inputs and outputs. The throughput and delay in IQ switch are heavily dependent on this scheduling decision. In past there has been a lot of research done to design good scheduling algorithms for IQ switches [1-3],[8]. In these studies there is 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 the cell-based systems, which only deal with the fixed-size cells. Using fixed-size cells in the switch makes the implementation of the scheduling algorithm of the switch much easier compared to the variable-length packets, but the following are the two main disadvantages with fixed-sized cell approach: (i) Packets arriving at input side need to be segmented into cells, requiring a special input segmentation module; and at the output side these cells need to be re-assembled. This induces significant implementation overhead. (ii) Packets may generate incomplete cells because a cell should not contain data belonging to two different packets. This can result in significant bandwidth loss. For example, if cell size is 64 bytes and packet size is 40 bytes then the amount of bandwidth lost is $24/64 \approx 37\%$! This motivates the study of packet scheduling algorithms.

The packet-based algorithms have been studied before in [7]. It is important to first understand the throughput region for the case of packet-scheduling algorithms. Naturally there is some similarity between packet-based and cell-based scheduling. For cell-based scheduling it is known that Maximum Weight Matching (MWM) algorithm is stable [1-6] for any admissible traffic. In [7] it is shown that canonical modification of the cell-based MWM for packet-based, which we denote as PB-MWM, achieves 100% throughput for any admissible Bernoulli i.i.d. traffic with packet lengths being bounded (rather probabilistically bounded and independent). In this paper we first study the PB-MWM algorithm. We study the throughput properties of the PB-MWM algorithm under general admissible traffic rather than restricting to the Bernoulli i.i.d. case. We first show that the PB-MWM is stable even for any form of re-generative admissible traffic, with the time of regeneration being finite in mean (note: Bernoulli i.i.d. traffic is special case of regenerative traffic). We obtain this result using a different proof technique, which seems to be somewhat simpler. Next we consider general admissible traffic with Strong Law of Large Numbers property. We show that there exists a counter example for which the PB-MWM is not stable. In general, this counter example shows that any scheduling algorithm that tries to schedule in work-conserving

0-7803-7753-2/03/$17.00 (C) 2003 IEEE

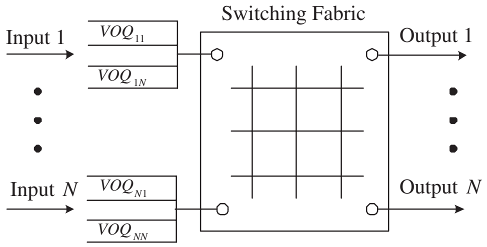

Fig. 1. An input-queued switch.

fashion or in maximal-sense every time, will not be stable. This counterexample suggests a fundamental difference between packet-based and cell-based scheduling algorithms. Hence in general to obtain stability we need to design a different type of packet-based scheduling algorithm. We propose a new class of algorithms which is called "waiting" algorithms. In particular, we show that "waiting" modification of PB-MWM is stable for any admissible traffic with bounded packet lengths using fluid technique similar to [6] (note: in general mean packet length should be bounded). The structure of this paper is as follows. In section II, we describe the input-queued switch architecture, the cell-based maximum weight matching (MWM) algorithm and the fluid model for the switch briefly. In Section III, the packet-based switching algorithms are defined. The canonical extension of MWM algorithm for the packet-based scenario is defined. In section IV, we prove the stability of the packet-based MWM for re-generative admissible traffic extending the proof of [7]. In section V we present the counter-example that motivates the classification of packet-based algorithm into two classes of "waiting" and "non-waiting" groups. In section VI, we introduce a simple waiting algorithm, which is proved to be stable using fluid techniques. Finally, in section VII we conclude the paper.

II. Input-Queued Switch

In this section we describe the model of an Input Queued (IQ) switch that is the main architecture studied in this paper.

Figure 1 shows the logical structure of an IQ switch. Although it is not necessary, we assume that the switch has the same number of input and output ports denoted by N. In fact, in practical designs, generally one input and output interface reside on the same "line card", thus the number of inputs and outputs is the same. We assume that the time is slotted and at each time slot, at most one data unit (of known fixed size) can arrive to each input port. We call this data unit a "cell". Cells arriving at input i and destined for output j are stored at input in a FIFO buffer called "virtual output queue" (VOQ), denoted here by VOQ_{ij}. This queue separation avoids the loss of throughput due to the head-of-line blocking problem. The cross-bar fabric is assumed to be memory-less. 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 "scheduling algorithm" decides which cells should be transferred between the inputs and outputs of the switch at every time slot, *i.e.*, it selects a matching between inputs and outputs in such a way that no input (respectively, output) may be matched to more than one output (respectively, input). We say a scheduling algorithm is *"work conserving"* or *"maximal"*, if an input is never left un-matched when it has a packet for an unmatched output.

We represent a matching by a $N \times N$ matrix $\mathbf{m} = [m_{ij}]$ where if input i is connected to output j, we have $m_{ij} = 1$, otherwise $m_{ij} = 0$. The set of all possible matchings is denoted by $\mathcal{M}$.

Let $A_{ij}(n)$ denote the number of cells that have arrived at input i destined for 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), that is 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}. This assumption on the arrival process is very mild.

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

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

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

Let $D_{ij}(n)$ show 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 called *"stable"* (rate stable) if, with probability one,

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

for any admissible arrival process $\mathbf{A}(n) = [A_{ij}(n)]$ with rate λ_{ij}.

We say the traffic is i.i.d. if the arrival process is such that, (a) the arrivals to different input ports are independent, and (b) the arrival to the same input port at different time slots are also independent. We would like to note that, a general admissible traffic, satisfying SLLN as above, does not need to have independence.

Let $Z_{ij}(n)$ show the number of cells in VOQ_{ij} at time n, including any arrival at time n, then the matrix $\mathbf{Z}(n) = [Z_{ij}(n)]$ shows the queue occupancy at time n. 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.

A. Maximum Weight Matching (MWM) Algorithm

At each time slot, MWM algorithm will select the matching with the maximum weight among all matchings in $\mathcal{M}$. If there are multiple such matchings, one of them is selected arbitrarily. We denote the maximum weight matching and its corresponding weight at time n by $\mathbf{m}^\star$ and $W^\star(n)$ respectively. That is,

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

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

In [1][3], it was shown that under any admissible Bernoulli i.i.d. traffic, MWM algorithm is stable. In [6] using the fluid model analysis it was shown that MWM is stable for any admissible traffic satisfying (1). The notion of stability (rate stability) in [6] is weaker than the notion of stability used in [1]-[3], but in [6] the stability is proved for a larger class of arrival traffic. In this paper also we adopt notion of rate stability as in (4).

B. Fluid Model and Switch Dynamics

This section describes the fluid model of a 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 non-decreasing function with respect to n. For a discrete-time switch the following three equations govern the dynamics of the switch:

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

$$\begin{aligned} D_{ij}(n) &= \sum_{\mathbf{m}\in\mathcal{M}} m_{ij} 1_{(Z_{ij}>0)} \left(T_{\mathbf{m}}(n) - T_{\mathbf{m}}(n-1)\right) \\ &+ D_{ij}(n-1), \end{aligned} \tag{9}$$

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

The first equation simply states that the number of cells in VOQ_{ij} equals the total number of arrivals minus the total number of departures. The second equation shows how to obtain the number of departure by considering all the matchings that can connect the input i to output j. The third equation simply states that at each time slot, exactly one of the possible matchings is used.

In [6], the fluid model of a discrete-time switch was introduced. We will use this fluid model in this paper without presenting any proofs or justification. An interested reader can refer to [6] for an elaborate exposition to this topic. From [6], the continuous equations governing the dynamics of the fluid model of switch described above are as follows. For every $i, j = 1, \ldots, N$,

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

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

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

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 $Z_{ij}(n)$.

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

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 renormalized by dividing by r and we let $r \to \infty$.

III. Packet-Based Switching

We described the structure of a cell-switch in the previous section, now we can define how a packet-based switch performs. Packets with different sizes can arrive to the switch. However we assume that the fabric works on fixed-size data units (cells). So in each time slot only one cell can be sent to each output. Thus, the received packets must be segmented into an integer number of cells. For simplicity, without loss of generality we will assume that each packet is made up of an integer number of cells. We constrain the scheduling algorithm to deliver contiguously all the cells obtained from the segmentation of the same packet, *i.e.*, at the output they are not interleaved by the cells from another input port. More formally we can define a packet-based scheduling algorithm as follows:

Definition 3: 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.

With this constraint of scheduling algorithm being non-preemptive on packets 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 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 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). Similar reasoning was given in [7] by authors in the favor of packet scheduling.

It is easy to convert a known cell-based algorithm into a packet-based one. Let us consider any cell-based scheduling algorithm $\mathcal{X}$ (*e.g.*, MWM, maximal matching, etc.). We can easily convert $\mathcal{X}$ into a packet-based algorithm as follows:

At each time slot, we divide the input-output ports into two disjoint sets:

1) *Busy ports*: the set of input-output ports which have been matched to each other in the previous time slot and are still in the middle of sending a packet.
2) *Free ports*: the set of input-output ports which either have no packets to send, or just finished sending a packet.

The scheduling algorithm PB-$\mathcal{X}$ keeps the matching already used by busy ports and finds a new (sub-)matching for free ports using the cell-based scheduling algorithm $\mathcal{X}$. Initially all the ports are assumed to be free.

In [7] Marsan et al. considered packet-based scheduling algorithms in the way defined as above. They described the model of a packet-based scheduling algorithm, and highlighted the effect of considering packet entity in designing the scheduling algorithms. They proved that the PB-MWM is stable for any admissible Bernoulli i.i.d. traffic. With the help of simulation results, they showed that packet-based scheduling algorithms could outperform a cell-based algorithm for certain cases. In the next section we give a different proof for stability of PB-MWM using the fluid model technique. This proof shows the stability of PB-MWM for a more general class of arrival process.

IV. PB-MWM Stability

In this section we provide a proof for stability of the packet-based MWM algorithm.

Definition 4: A matching $\mathbf{m}(n)$ used at time n is called *"k-imperfect"* if

$$\mathbf{m}(n) = \mathbf{m}^\star(n-k). \tag{16}$$

In other words, $\mathbf{m}$ is k-imperfect if it is equal to the maximum weight matching at k time slots ago.

Obviously, any maximum weight matching is a 0-imperfect matching at the time it is chosen by the scheduler. The following Lemma states a very simple but important property of k-imperfect matchings.

Lemma 1: The weight of a k-imperfect matching is at most $2kN$ different from the weight of the maximum weighted (0-imperfect) matching at any time slot, *i.e.*, if $\mathbf{m}(n)$ is a k-imperfect matching with weight $W_{\mathbf{m}}(n)$, used at time slot n, then;

$$W_{\mathbf{m}}(n) \geq W^\star(n) - 2kN. \tag{17}$$

Proof: For any matching $\mathbf{m}' \in \mathcal{M}$, note that $W_{\mathbf{m}'}(n-k)$ shows its weight at time $n-k$ and $W_{\mathbf{m}'}(n)$ shows its weight at time n. Then the following inequalities hold under any scheduling algorithm:

$$W_{\mathbf{m}'}(n-k) - kN \leq W_{\mathbf{m}'}(n) \leq W_{\mathbf{m}'}(n-k) + kN. \tag{18}$$

This is true because of the following simple reason: during k time slots, at most k cells can arrive (depart) at an input port, which in turn can increase (decrease) queue size at any input port by at most k. There are N input ports, and hence the net weight can increase (decrease) by at most kN.

We know that $\mathbf{m}(n)$ is k-imperfect, thus, $\mathbf{m}(n) = \mathbf{m}^\star(n-k)$, *i.e.*, at time $n-k$, $\mathbf{m}$ has the largest weight among all possible matchings;

$$\forall\, \mathbf{m}'' \in \mathcal{M} \quad W_{\mathbf{m}(n)}(n-k) \geq W_{\mathbf{m}''}(n-k). \tag{19}$$

Thus if we select $\mathbf{m}'' = \mathbf{m}^\star(n)$ we get;

$$W_{\mathbf{m}(n)}(n-k) \geq W_{\mathbf{m}^\star(n)}(n-k). \tag{20}$$

Rewriting (18) for $\mathbf{m}' = \mathbf{m}(n)$ and $\mathbf{m}' = \mathbf{m}^\star(n)$, we get;

$$W_{\mathbf{m}(n)}(n) + kN \geq W_{\mathbf{m}(n)}(n-k). \tag{21}$$

$$W_{\mathbf{m}^\star}(n-k) \geq W_{\mathbf{m}^\star}(n) - kN. \tag{22}$$

Combining (20), (21), and (22) we obtain the result stated in lemma 1. ■

Let us consider the following scheduling algorithm, which we denote as $\mathcal{S}$:

1) Let $\mathbf{s}(n)$ be the schedule used by $\mathcal{S}$ at time n.
2) At time $n+1$
   a) if all ports are free then use $\mathbf{s}(n+1) = \mathbf{m}^\star(n+1)$.
   b) else set $\mathbf{s}(n+1) = \mathbf{s}(n)$.

Let T be the time between two successive occurrences of the event that all ports are free. Note that the matching obtained by algorithm $\mathcal{S}$ is at worst T-imperfect, by definition, and T is a random variable which depends on the arrival process and the packet lengths. We assume that the packet lengths are bounded and the arrival process is stationary. Let $p_T(t)$ denote stationary probability of event $\{T = t\}$ and $W_{\mathcal{S}}(n)$ show the weight of matching obtained by scheduling algorithm $\mathcal{S}$ at time n. Then,

$$\begin{aligned}\mathbb{E}\{W_{\mathcal{S}}(n)|\mathbf{Z}(n)\} &\geq \sum_{t=0}^{\infty} p_T(t)\,[W^\star(n) - 2tN] \\ &= W^\star(n) - 2N\sum_{t=0}^{\infty} t p_T(t).\end{aligned} \tag{23}$$

Thus,

$$\mathbb{E}\{W_{\mathcal{S}}(n)|\mathbf{Z}(n)\} \geq W^\star(n) - 2N\mathbb{E}(T). \tag{24}$$

If $\mathbb{E}(T)$ is finite we say that the traffic pattern is *"regenerative"*. In other words, it has property that on average it requires a finite amount of time to reach the state where all the input and output ports are free. Note that this property is related to the traffic pattern and not to the scheduling algorithm $\mathcal{S}$. It is already shown in [7] that if the traffic is formed by variable length packets with independent random size (with finite average and variance), and if it is admissible Bernoulli i.i.d. traffic, then the average value of T is bounded. In general, it is not required that the traffic be Bernoulli i.i.d. so that the regenerative property holds true. There is a much larger class of distributions under which we get this property. For all regenerative traffic patterns, we prove the stability of algorithm

$\mathcal{S}$. The following is a key lemma, which states a general result about stability.

Lemma 2: A scheduling algorithm is rate-stable for any admissible traffic which satisfies (1), if the average value of the weight of the matching it uses at each time slot, is at most away from the maximum weight by a bounded constant value, *i.e.*, if

$$\mathbb{E}\{W(n)|\mathbf{Z}(n)\} \geq W^{\star}(n) - B, \tag{25}$$

then the algorithm is stable.

Proof: Let $\tilde{\mathbf{Z}}(t) = \left[\tilde{Z}_{ij}(t)\right]$ and $\tilde{\mathbf{D}}(t) = \left[\tilde{D}_{ij}(t)\right]$, then consider the Lyapunov function $L(t)$ defined as,

$$L(t) = \left\langle \tilde{\mathbf{Z}}(t), \tilde{\mathbf{Z}}(t) \right\rangle = \sum_{i,j} \tilde{Z}_{ij}^2(t). \tag{26}$$

It was shown in [6] that for MWM, $\dot{L}(t) < 0$ if any $\tilde{Z}_{ij} > 0$. This in turn implies that if $\tilde{\mathbf{Z}}(0) = 0$ then $\tilde{\mathbf{Z}}(t) = 0$ for all t. This proves the rate-stability of the switch. Hence, if we show that $\dot{L}(t) < 0$ if any $\tilde{Z}_{ij} > 0$ for any scheduling algorithm in consideration with property that the expected weight of the matching used is at most a bounded constant away from the weight of MWM, the rest of the proof for the rate-stability follows from [6].

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

$$\begin{aligned} \frac{\partial L(t)}{\partial t} &= 2\left\langle \frac{\partial \tilde{\mathbf{Z}}(t)}{\partial t}, \tilde{\mathbf{Z}}(t) \right\rangle \\ &= 2\left\langle \mathbf{\Lambda} - \frac{\partial \tilde{\mathbf{D}}(t)}{\partial t}, \tilde{\mathbf{Z}}(t) \right\rangle \\ &= 2\left\langle \mathbf{\Lambda}, \tilde{\mathbf{Z}}(t) \right\rangle - 2\left\langle \tilde{\mathbf{Z}}(t), \frac{\partial \tilde{\mathbf{D}}(t)}{\partial t} \right\rangle. \end{aligned} \tag{27}$$

Substituting (12) we obtain;

$$\begin{aligned} \left\langle \tilde{\mathbf{Z}}(t), \frac{\partial \tilde{\mathbf{D}}(t)}{\partial t} \right\rangle &= \sum_{\mathbf{m} \in \mathcal{M}} \left\langle \tilde{\mathbf{Z}}(t), \mathbf{m} \right\rangle \frac{\partial \tilde{T}_{\mathbf{m}}}{\partial t} \\ &= \sum_{\mathbf{m} \in \mathcal{M}} \tilde{W}_{\mathbf{m}}(t) \frac{\partial \tilde{T}_{\mathbf{m}}}{\partial t}, \end{aligned} \tag{28}$$

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

Let us define $\Delta(n)$ as the difference between the weight of the MWM and 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,

$$\tilde{\Delta}(t) = \lim_{r \to \infty} \frac{\hat{\Delta}(rt)}{r} \leq \lim_{r \to \infty} \frac{\hat{B}}{r} = 0. \tag{29}$$

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

$$\tilde{W}_{\mathbf{m}}(t) = \tilde{W}^{\star}(t). \tag{30}$$

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

$$\begin{aligned} \left\langle \tilde{\mathbf{Z}}(t), \frac{\partial \tilde{\mathbf{D}}}{\partial t} \right\rangle &= \sum_{\mathbf{m} \in \mathcal{M}'} \tilde{W}^{\star}(t) \frac{\partial \tilde{T}_{\mathbf{m}}(t)}{\partial t} \\ &= \tilde{W}^{\star}(t) \sum_{\mathbf{m} \in \mathcal{M}'} \frac{\partial \tilde{T}_{\mathbf{m}}(t)}{\partial t}. \end{aligned} \tag{31}$$

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

$$\sum_{\mathbf{m} \in \mathcal{M}'} \tilde{T}_{\mathbf{m}}(t) = t, \tag{32}$$

changing $\mathbf{m} \in \mathcal{M}$ to $\mathbf{m} \in \mathcal{M}'$. Now combining this result with (31) we obtain

$$\left\langle \tilde{\mathbf{Z}}(t), \frac{\partial \tilde{\mathbf{D}}}{\partial t} \right\rangle = \tilde{W}^{\star}(t). \tag{33}$$

Hence, from (27) the derivative of $L(t)$ will be

$$\frac{\partial L(t)}{\partial t} = 2\left\langle \mathbf{\Lambda}, \tilde{\mathbf{Z}}(t) \right\rangle - 2\tilde{W}^{\star}(t). \tag{34}$$

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

$$\mathbf{\Lambda} \preceq \sum_{k=1}^{K} \gamma_k \mathbf{m}_k, \quad \sum_{k=1}^{K} \gamma_k < 1, \tag{35}$$

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

$$\left\langle \tilde{\mathbf{Z}}(t), \mathbf{m}_k \right\rangle \leq \tilde{W}^{\star}(t). \tag{36}$$

Combining (35), (34), (36), we obtain;

$$\begin{aligned} \frac{\partial L(t)}{\partial t} &\leq 2\left\langle \tilde{\mathbf{Z}}(t), \sum_{k=1}^{K} \gamma_k \mathbf{m}_k \right\rangle - 2\tilde{W}^{\star}(t) \\ &= 2\sum_{k=1}^{K} \gamma_k \left\langle \tilde{\mathbf{Z}}(t), \mathbf{m}_k \right\rangle - 2\tilde{W}^{\star}(t) \\ &\leq 2\left(\sum_{k=1}^{K} \gamma_k - 1\right) \tilde{W}^{\star}(t) \\ &\leq 0. \end{aligned} \tag{37}$$

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. ■

Now combining this lemma with (24), we conclude that the algorithm $\mathcal{S}$ is stable as long as $\mathbb{E}(T) < \infty$. Note that, the proof does not require the bounded packet lengths condition, but requires only independent packet lengths with bounded mean. Hence, *Theorem 1:* Algorithm $\mathcal{S}$ is stable under regenerative admissible input traffic.

We would like to note that, under PB-MWM algorithm the time between successive occurrences of event when all ports become free will also have the required property, *i.e.*, under Bernoulli i.i.d. traffic for independent packet lengths with bounded mean, again $\mathbb{E}(T) < \infty$. Hence the stability for PB-MWM will again follow from Lemma 2, and . This shows that as proved for $\mathcal{S}$, the PB-MWM algorithm is also stable under regenerative admissible traffic, which is more general than the Bernoulli i.i.d. traffic.

Theorem 2: PB-MWM Algorithm is stable under regenerative admissible input traffic.

In the next section we show that there are still admissible traffic patterns for which the PB-MWM algorithm is unstable.

V. Packet-Based Algorithm Classification

It is proved in [6] that cell-based MWM algorithm has strong stability property that it is stable as long as the input traffic is admissible and property (1) holds. It does not require any other condition on distribution. In previous section we proved the stability for PB-MWM (and $\mathcal{S}$) for admissible traffic with additional condition that it should be regenerative. The question that arises is: whether the PB-MWM (or $\mathcal{S}$) is stable for all admissible input traffics which only satisfy (1). We show that the answer is no, using a simple counter-example.

Consider a switch operating under PB-MWM (or $\mathcal{S}$) with input traffic pattern as shown in Figure 2. A_{ij} $(i, j = 1, 2)$ shows the arrival to VOQ_{ij}. The traffic pattern is periodic with period equal to 10. Note that no input or output is overloaded. In fact $\lambda_{1,1} = 0.8$, $\lambda_{1,2} = 0.1$, $\lambda_{2,1} = 0.1$,and $\lambda_{2,2} = 0.8$. The switch can use one of the two possible matchings, namely $\mathbf{m}_1$ which is called the parallel matching and $\mathbf{m}_2$ the cross matching, *i.e.*,

$$\mathbf{m}_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad \mathbf{m}_2 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}. \qquad (38)$$

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 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. Hence both input ports are never free together. 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.

The counter-example described above also shows that any work-conserving or maximal algorithm is not stable for that particular traffic pattern. This motivates us to classify the packet-scheduling algorithm in the following two classes:

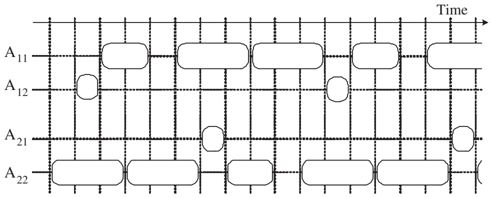

Fig. 2. Traffic pattern.

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

The above counter-example suggests the following general result about the work-conserving (maximal) algorithms.

Theorem 3: There is no work-conserving packet-based scheduling algorithm that is stable under any admissible traffic (satisfying condition (1)).

Note that even if an algorithm *wait*s for finite number of time slots it becomes work-conserving after some time and hence applying a traffic similar to Figure 2 after that time, will make it unstable.

VI. A Generally Stable Waiting Packet-Based Algorithm

In this section we describe a waiting algorithm. We will show that the waiting-MWM algorithm will achieve 100% throughput for any admissible traffic pattern and in particular, it will be stable for the traffic pattern described in previous section for which PB-MWM or any packet-based work-conserving scheduling policy was unstable.

The waiting algorithms are motivated from the counter-example described in 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 Figure 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 schedule to the weight of MWM schedule. Hence if waiting is done frequently enough then the weight of schedule is always not more than a bounded constant away from MWM, by reasoning similar to Lemma 1. However note that waiting means that during the waiting period some ports are losing bandwidth. Hence if waiting is done too aggressively then the algorithm can not utilize full bandwidth. These observations lead to the following waiting algorithm which we denote as PB-wMWM.

A. PB-wMWM

The switch runs at speedup $(1 + \epsilon)$ for arbitrarily small positive constant $\epsilon > 0$.

Let the maximum length of any packet be L (this assumption can be relaxed to mean packet length being finite which will be described later). Divide the time into period of length $\frac{L}{\epsilon}$ units. Thus time is considered as $[0, \frac{L}{\epsilon}]$, $[\frac{L}{\epsilon}+1, 2\frac{L}{\epsilon}]$, and so on. Scheduling decisions are made only when any of scheduled packets finishes its service and corresponding ports get empty. Let one or more packets get served at time $n \in [k\frac{L}{\epsilon}+1, (k+1)\frac{L}{\epsilon}]$. Consider the following cases:

1) If $n \in [k\frac{L}{\epsilon}+1, (k+1)\frac{L}{\epsilon} - \frac{L}{1+\epsilon}]$ use usual PB-MWM to match the free ports as before.
2) Otherwise wait on all the packets till all scheduled packets get over and all ports are free. After that, use full MWM to re-schedule all the ports and serve.

Note that the above algorithm at most loses bandwidth of L per every $\frac{L}{\epsilon}$ time slots. That is, it loses bandwidth of ϵ per time slot at most. The algorithm runs at speedup $(1+\epsilon)$ in order to make up for this lost bandwidth. We state the following theorem about stability of PB-wMWM.

Theorem 4: The PB-wMWM algorithm is stable (rate stable) under any admissible traffic (with property (1)) at at speedup $(1+\epsilon)$ for any $\epsilon > 0$.

Proof: Note that the way algorithm is defined, every $\frac{L}{\epsilon}$ time the weight of matching is same as weight of maximum weight matching. Thus any time the algorithm is at worst $\frac{L}{\epsilon}$-imperfect. Hence by Lemma 1, the weight of the matching is at most $B_\epsilon = 2N\frac{L}{\epsilon}$ away from MWM. The fraction of time the algorithm idles on any of the ports is bounded above by

$$\frac{\frac{L}{1+\epsilon}}{\frac{L}{\epsilon}} = \frac{\epsilon}{1+\epsilon}. \tag{39}$$

Under speedup $(1+\epsilon)$ assuming the algorithm is scheduling all the time, the equation (13) changes to

$$\sum_{\mathbf{m}\in\mathcal{M}} \tilde{T}_{\mathbf{m}}(t) = (1+\epsilon)t. \tag{40}$$

But in our algorithm, since it is waiting, the above equation may not be true. From above discussion, at worst ϵ fraction of the bandwidth is lost in waiting. That is, at least

$$(1+\epsilon)\left(1 - \frac{\epsilon}{1+\epsilon}\right) = 1, \tag{41}$$

is the effective speedup obtained. Thus, the equation (13)of the fluid model changes to

$$\sum_{\mathbf{m}\in\mathcal{M}} \tilde{T}_{\mathbf{m}}(t) \geq t. \tag{42}$$

In other words,

$$\sum_{\mathbf{m}\in\mathcal{M}} \frac{\partial \tilde{T}_{\mathbf{m}}(t)}{\partial t} \geq 1. \tag{43}$$

Now the arguments similar to ones used in proof of Lemma 2, yield the desired result that PB-wMWM is stable. ■

The above algorithm PB-wMWM, assumes the packet lengths to be bounded and bound is known. But in reality the might not be known. Further we do not require the packet lengths to be bounded, but only mean packet length should be bounded. To address this issue we modify the PB-wMWM algorithm as follows.

B. Modified PB-WMWM (PB*-WMWM)

1) Initially start with the MWM algorithm and start waiting immediately.
2) Compute the maximum amount of idling done by any port. When the waiting starts, there are some unfinished packets. Note that the maximum waiting done by any port is at most the maximum length of any packet that was under schedule. Let $L_e(1)$ represent the maximum length of packets under schedule.
3) Set $M(1) = \frac{L_e(1)}{\epsilon}$ and do the PB-MWM for $M(1)$ time slots and then start waiting after that.
4) Now let $L_e(2)$ be the maximum length of the packets under schedule when the waiting starts at the end of $M(1)$ time slots.
5) Similarly define $M(2) = \frac{L_e(2)}{\epsilon}$.
6) Continue this process recursively over time. In general we obtain the following recursive expression,

$$M(l) = \frac{L_e(l)}{\epsilon}. \tag{44}$$

The two main properties required in the proof of Theorem 5 are: (a) The effective speed is at least 1, and (b) the weight of schedule used by algorithm is at most bounded constant away from the weight of MWM. In the above algorithm, let's compute these two quantities as follows:

(a) The effective bandwidth lost: In the l^{th} period the total idling per port is at most for time $\frac{L_e(l)}{\epsilon}$ while the length of period is

$$P(l) = M(l) + \frac{L_e(l)}{1+\epsilon} = \frac{L_e(l)}{\epsilon} + + \frac{L_e(l)}{1+\epsilon}. \tag{45}$$

Thus the fraction of bandwidth lost is at most

$$\begin{aligned} \frac{\frac{L_e(l)}{1+\epsilon}}{P(l)} &= \frac{\frac{1}{1+\epsilon}}{\frac{1}{\epsilon} + \frac{1}{1+\epsilon}} \\ &\leq \frac{\epsilon}{1+\epsilon}. \end{aligned} \tag{46}$$

Note that this bandwidth loss is same as the loss in PB-wMWM computed in proof of Theorem 4.

(b) The difference between the weight of MWM and the schedule will be at most $M(l)(1+\epsilon)$ that is,

$$\frac{L_e(l)(1+\epsilon)}{\epsilon}. \tag{47}$$

Given that $L_e(l)$ has bounded mean and packet lengths are independent we will obtain that the above quantity is bounded almost surely as required in Lemma 2.

From above discussion we obtain the following Theorem.

Theorem 5: The PB*-wMWM is rate-stable for any admissible traffic with property (1) and independent packet lengths with bounded mean.

VII. Conclusion

In this paper we considered the packet-scheduling algorithms for IQ switch architecture. The result of [7] showed that modification of cell-based MWM for packet scheduling yields 100% throughput for any admissible Bernoulli i.i.d. traffic with independent packet lengths of bounded mean. We generalized this result for some what broader class of arrival traffic pattern. We showed that there exists 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 fluid limit technique. It is interesting to note that the work-conservation for packet scheduling is not always beneficial in this sense, unlike cell-based scheduling. This suggests that scheduling packet-based is quite different from the cell-based scheduling.

Acknowledgment

The authors thank B. Prabhakar and N. McKeown for suggesting the problem and discussions.

References

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

[2] L. Tassiulas and A. Ephremides, "tability Properties of constrained queuing systems and scheduling plicies for maximum throughput in multihop radio networks," *IEEE Trans. Automatic Control*, vol. 37, no. 12, Dec 1992, pp. 1936-1948.

[3] N. McKeown, V. Anantharam, and J. Walrand, "Achieving 100% Throughput in an Input-Queued Switch," *IEEE Transaction on Comm.*, vol. 47, no. 8, Aug. 1999, pp. 1260-1267.

[4] N. McKeown, "iSLIP: a scheduling algorithm for input-queued switches," *IEEE Transaction on Networking*, vol. 7, no.2, April 1999, pp. 188-201.

[5] N. McKeown, "Scheduling algorithms for input-queued cell switches," *PhD Thesis*, University of California, Berkeley, May 1995.

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

[7] MA. Marsan, A. Bianco, P. Giaccone, E. Leonardi, and F. Neri, "Packet Scheduling in Input-Queued Cell-Based Swithces," *INFOCOM 2001*, pp. 1085-1094.

[8] P. Giaccone, B. Prabhakar,and D. Shah, "Towards Simple, High-Performance Schedulers for High-aggregate bandwidth Switches," *INFOCOM 2002*,New York, NY.