EE384Y: Packet Switch Architectures II

# Cell Switching vs. Packet Switching

*Abtin Keshavarzian & Yashar Ganjali*

Department of Electrical Engineering
Stanford University

June 2002

## Abstract

In current switching systems, all packets are divided into equal-sized cells before being scheduled. This is because of the fact that cell-based scheduling is easier to be implemented in hardware. In this paper, we consider the case in which the switch does not split packets and the scheduling is done over the whole packets. In this paper, we classify packet-based scheduling algorithms into *waiting* and *non-waiting* and show that any algorithm in the second class is not generally stable. Then we show that for a given packet-based scheduling algorithm, which we call PB-MWM, the switch is stable under i.i.d. traffic assumption. Finally, we compare cell-based Maximum Weighted Matching and our PB-MWM algorithm. We show that even though these two seem to be very different in structure, they perform similarly.

**Keywords:** Packet switching, scheduling, variable length packets.

## 1 Introduction

When designing any switching system, there are two important factors to be considered, namely the overall throughput of the system and the delay of packets. These two factors are tremendously affected by the scheduling algorithm used by the switch and have been studied extensively for input-queued, output-queued, and combined-input-output-queued architectures [1, 2, 3, 4]. In all 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 switching is done, the cells are reassembled in the form of the original packet before leaving the system.

In contrary to this common assumption, we can 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 fixed-size cells. Obviously, using fixed-length cells in the switch makes the hard-

ware implementation much easier than using variable-length packets, but the question is can we gain a better performance (delay, throughput, ...) using packet-based systems. In this paper, we will try to investigate this problem, and study the trade-off between packet-based and cell-based switching systems.

The structure of this paper is as follows: in Section 1.1 we will state the problem in more detail. Then, in Section 2 we briefly review previous results in this area. In Section 3 we propose a number of packet-based scheduling algorithms. We classify all packet-based scheduling algorithms into two classes of *waiting* and *non-waiting* algorithms and show that no non-waiting algorithm can gain 100% throughput under generally admissible traffic. In section 3.2 we prove that there is a non-waiting packet-based scheduling algorithm which gains 100% throughput under i.i.d. traffic. In section 4 we compare the performance of packet-based and cell-based scheduling algorithms using simulation. Finally we conclude in Section 5.

## 1.1 Problem Statement

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 delay observed by the last cell of any packet. Therefore, a scheduling algorithm which switches the last cell of a packet with a large delay is considered to perform poorly, even if it performs very 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).

Obviously, a possible drawback for packet-based switching systems is that long packets may keep a pair of input-output ports busy, therefore increasing the delay of other small packets waiting for the same output port.

Based on the above observations, a very interesting problem is the study of the trade-off between cell-based and packet-based scheduling algorithms and the role of traffic pattern in the performance of each system.

McKeown et. al have shown that using Maximum Weighted Matching (MWM) as a scheduling algorithm for a cell-based switch results in 100% throughput for i.i.d. traffic pattern [2]. Prabhakar and Dai have shown that the stability holds for any admissible traffic [3]. There are also a number of cell-based scheduling algorithms which are stable under any admissible i.i.d. traffic even though they cannot achieve 100% throughput in general. There are no similar results for packet-based scheduling algorithms which we are aware of.

In this paper we will propose a number of algorithms and study their stability under different traffic patterns.

## 1.2 Preliminaries

Let us start with formally defining what a packet-based scheduling algorithm is. Here we assume that a cell-based switching system is an input-queued crossbar which also implements virtual output queues. Know-

ing this structure, we can define a packet-based switching system as follows.

**Definition 1**. A *packet-based scheduling algorithm* for a switch is a cell-based scheduling algorithm such that once it starts transferring a packet from an input port to an output port it does not stop until the whole packet is completely transfered to the corresponding output port.

As we can see in this definition, if we assume that all packets are of the same size a packet-based scheduling algorithm will be the same as a cell-based scheduling algorithm.

# 2 Previous work

Marsan et. al [5] have considered this problem and have proposed a simple scheduling algorithm for Input-queued switches. They use simulations to show the efficiency of their method and do not provide any analytical proofs. They show that in the case of packet-based switching systems, input queuing architectures can provide advantages over output-queuing architectures.

Manjunath and Sikdar [6] give an analytic model for the delay in an input-queued, packet-based switches under Poisson and self similar traffic patterns. They validate their model using simulations. Based on their model, they show that a first-come-first-served service in the virtual output queued system gives the least average delay. Their model can be easily extended to consider priorities in the input queue, but unfortunately, their model does not address many architectures that are possible for packet-based switching systems.

# 3 Main results

In this section we introduce different methods for converting cell-based scheduling algorithms into packet-based algorithms and study the properties of the proposed methods.

## 3.1 Proposed Algorithms

One way to convert any cell-based scheduling algorithm to a packet-based one, is to assume that we have got large cells such that any packet can be encapsulated in a cell. We call this method algorithm 1. Obviously we are wasting a lot of bandwidth in this method. One might think that by taking the cell size to be the maximum of the packets which are being transfered we can gain some advantage. This is called algorithm 2. As we will see later, both of these algorithms perform very poorly in general.

Let us consider a more complex method. We consider any cell-based scheduling algorithm $X$ (e.g. MWM, maximal matching, etc.). We can easily convert $X$ into a packet-based algorithm as follows.

**PB-$X$ scheduling algorithm:** At each time slot $t$ 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-$X$ keeps the matching already used by busy ports

and finds a new (sub-)matching for free ports using the cell-based scheduling algorithm $X$ (Figure 1).

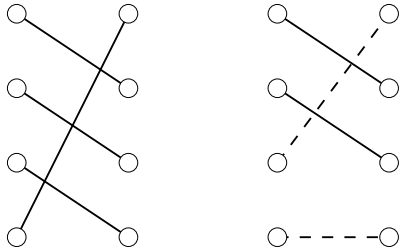


Figure 1: **a)** The matching used by the PB-$X$ algorithm at time $t$. **b)** The solid line are the input-output ports which are still busy. PB-$X$ recomputes a matching for the free ports indicated by the dashed lines.

### 3.1.1 Non-waiting vs. waiting packet-based scheduling algorithms

In this section we will classify packet-based algorithms into two classes called *non-waiting* and *waiting* algorithms and study their properties.

**Definition 2**. A packet-based scheduling algorithm PBA is said to be *non-waiting* if PBA starts transferring packets from input port $i$ to output port $j$ whenever there is a packet at $VOQ_{ij}$ and neither the input port $i$ nor the output port $j$ is busy.

Intuitively, a non-waiting scheduling algorithm never defers the transfer of a (head of line) packet unless one of the input or output ports corresponding to that packet are busy. On the other hand we can have algorithms that may defer transferring a packet even though none of the corresponding input or output ports are busy. We call such algorithms *waiting* algorithms..

Since any PB-X algorithm finds a matching for all free input-output ports at any time slot, any PB-X scheduling algorithm is a non-waiting algorithm.

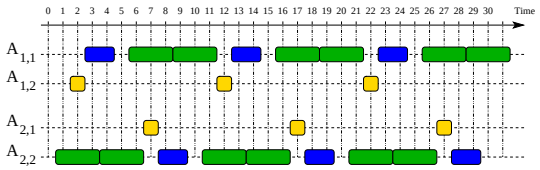


Figure 2: Arrival pattern which makes the switch unstable.

With the help of a counter-example we can easily show that no non-waiting packet-based scheduling algorithm can be generally stable. Consider a 2x2 switch and assume that packets of length 1,2, or 3 arrive to this switch according to the pattern shown in Figure 2. $A_{i,j}$ shows the packets arrived at input port $i$ going to output port $j$. We note that the traffic is strictly admissible (no input or output is over-subscribed). When the first packet arrives (packet with length 3 to $A_{2,2}$) the switch selects the parallel matching (1⇒1 and 2⇒2) and is forced to keep it for the next three units of time. But before it can switch again, a packet of size 2 arrives to $A_{1,1}$ and force the switch to keep the same matching. It can be seen that the same process occurs alternatively between lines $A_{11}$ and $A_{22}$ forcing the system to keep the same matching forever. Therefore, packets of size 1 can never leave the switch and thus the switch is not stable.

## 3.2 PB-MWM under i.i.d. traffic

In Section 3.1.1 we proved that any PB-X algorithm is not stable under generally admissible traffic. In this section, we show that under i.i.d. input traffic, there is a class of algorithms, which include PB-MWM that can gain 100% throughput. To show this let us start with the following definition.

**Definition 3**. A matching $M$, between input and output ports, is called a *$k$-imperfect matching* at time slot $t$, if the scheduling algorithm of a switch chooses $M$ as a maximum weighted matching at time $t-k$ and the switch continues to use $M$ during the time interval $[t-k, t]$.

Obviously, any maximum weighted 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**. *If $W^\star$ is the weight of any maximum weighted matching at time $t$ and $W$ is the weight of a $k$-imperfect matching at that time, $W \geqslant W^\star - 2Nk$.*

**Proof.** Let $M$ be the maximum weighted matching used by the scheduler at time $t-k$ and $M^\star$ be a maximum weighted matching at time $t$. For any input-output pair in $M$ we can have at most $k$ departures and therefore we have at most $Nk$ departures if the system uses $M$ for $k$ consecutive time slots. Also we have at most $Nk$ arrivals to the input-output pairs in $M^\star$ during these $k$ time slots (Figure 3). Therefore the weight of $W$ is at least $W^\star - 2Nk$. ■

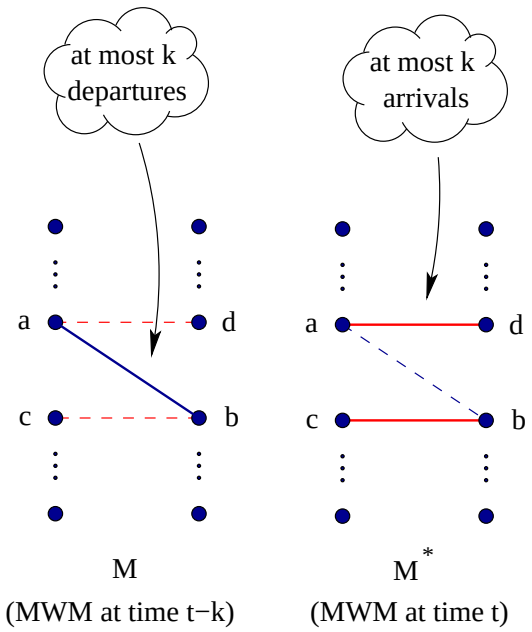


Figure 3: $M$ is the matching used by the switch for the time interval $[t-k, t]$ and $M^\star$ is the MWM at time $t$. During the time interval $[t-k, t]$ we can have up to $Nk$ arrivals to $M^\star$ and up to $Nk$ departures from $M$.

**Theorem 1**. *There is a class of PB-X algorithms which are stable under i.i.d. input traffic.*

**Proof.** Let us consider any scheduling algorithm SA such that whenever all input-output pairs are free, SA uses a maximum weighted matching and keeps the same matching (or a matching with a greater weight) till the time when all input-output pairs are free. We say that such an algorithm is in state $i$ $(0 \leqslant i)$ at time $t$ if the algorithm has updated its matching at time $t-i$ for the last time. Based on the way we have defined the states, we can have only transitions between states $i$ and $i+1$ (for $0 \leqslant i$) or from state $i$ to state 0. Let us

assume that the probability of going from state $i$ to state $i+1$ is $P_i$ and the probably of going back to state 0 from state $i$ is $Q_i$ (Figure 4). Obviously, $P_i + Q_i = 1$.

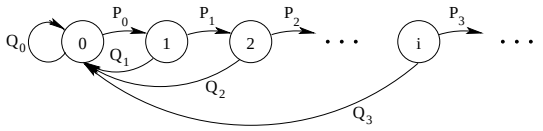


Figure 4: The structure of states in the SA algorithms.

If for every $i > 0$ we have $Q_i > \alpha$ for a given $\alpha > 0$, then we can to verify that the probability of being in state $i$, denoted by $\pi_i$, in steady state is at most $\alpha^{i+1}$. Using Lemma 1 we also know that if the system is in state $i$, we have $W_i \geqslant W^\star - 2Ni$, where $W_i$ is the weight of the matching which is being used by the switch and $W^\star$ is the weight of the maximum weighted matching at that time.

Now, let us compute the expected weight of the matchings used by the switch.

$$\begin{aligned} E\{W\} &= \sum_{i \geqslant 0} W_i P_i \\ &\geqslant \sum_{i \geqslant 0} W^\star - 2NiP_i \\ &\geqslant W^\star \sum_{i \geqslant 0} P_i - 2N \sum_{i \geqslant 0} P_i \\ &\geqslant W^\star - 2N \sum_{i \geqslant 0} i\alpha^{i+1} \end{aligned}$$

Since the arrivals are i.i.d. we can easily show that such an alpha always exists, because at any time slot the probability that all input-output lines become free is not zero (even though this probability may be very small). Therefore, we have $E\{W\} \geqslant W^\star - C$ which proves that the system is stable. ■

Since the weight of the matchings used by the PB-MWM is always greater than or equal to the weights of the matchings used by the SA algorithm in the previous proof, we can easily derive the following corollary.

**Corollary 1**. *The PB-MWM scheduling algorithm is stable under i.i.d. traffic.*

# 4 Simulation results

In this section, we compare the performance of packet-based scheduling algorithms and cell-based scheduling algorithms. We use a simulation program to obtain an estimate of the average delay seen by the packets.

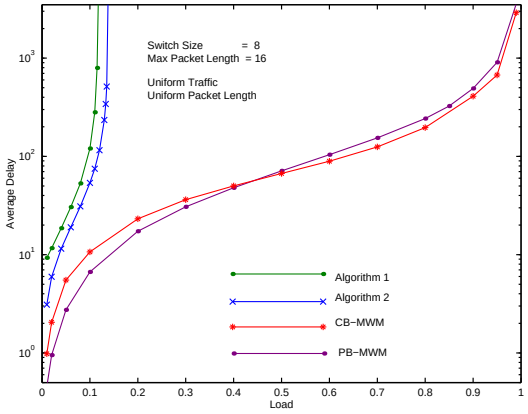


Figure 5: Average delay vs. input traffic load (uniform traffic, uniform packet length)

Figure 5 shows the average delay seen by packets versus the traffic load for four different algorithms. An 8x8 switch with maximum packet length of 16, uniform input traffic and uniform packet length distribution is considered. The curves corresponding to algorithms 1 and 2 (Section 3.1) show that these algorithms perform very poorly, even for low traffic loads.

The two other curves correspond to packet-based and cell-based MWM algorithms. For the cell-based case, in the simulation process, the delays seen by each packet (and not by cells) is considered, i.e., the delay assigned to a packet is the delay which is observed by the last cell of a packet.

It is worth noticing that although these two algorithms are completely different, they have pretty close delay curves. However, for small load values, the packet-based MWM performs slightly better than the cell-based algorithm, and the situation is reversed for higher loads. The rate at which the two curves get identical average delay is about 0.5 in this particular example.

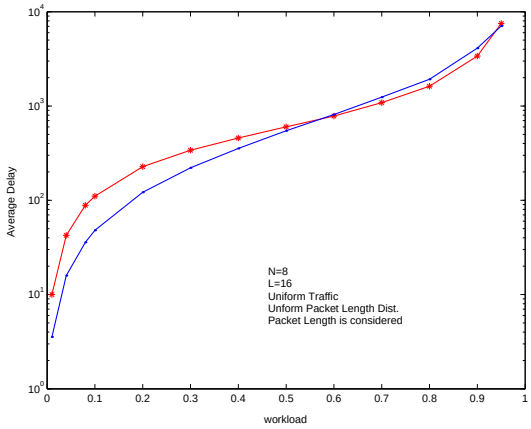


Figure 6: Weighted average delay vs. input traffic load (uniform traffic, uniform packet length)

To compute the average delay in the previous plot, we took the average without considering the length of each packet. In other words, each packet has the same effect on the total average no matter how long the packet is. However, it seems reasonable to consider the packet lengths in computing the average delay, i.e., to multiply the delay seen by each packet by its length and take the average over all packets. In Figure 6 we can see the average delays obtained with this modification, for a switch with the same specification as in Figure 5. Again, we see that these two curves are pretty close to each other, and for low traffic loads the packet-based algorithm performs better, while for higher loads the cell-based algorithm has better delays on average.

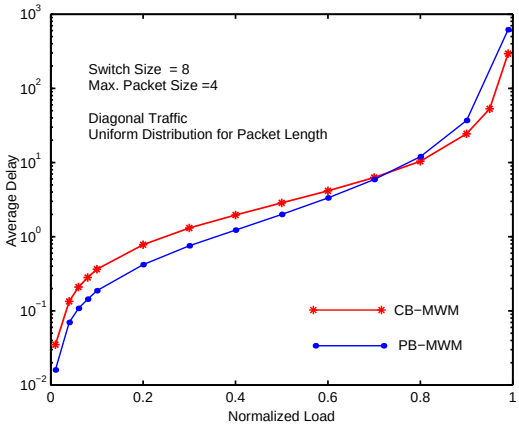


Figure 7: Average delay vs. input traffic load (diagonal traffic, uniform packet length)

In Figure 7 we examine the delay performance of the switch for a non-uniform traffic pattern. The traffic applied to the switch has diagonal distribution, i.e., $\lambda_{i,i} = 0.9$ and $\lambda_{i,i+1} = 0.1$ and all other elements are zero in the rate matrix. We again see the same behavior as before, but the crossing point of the two curves is now somewhere around 0.7.

In Figure 8 we assume a different distribution for packet lengths. The traffic pattern is uniform but the length of the packets has the following distribution:

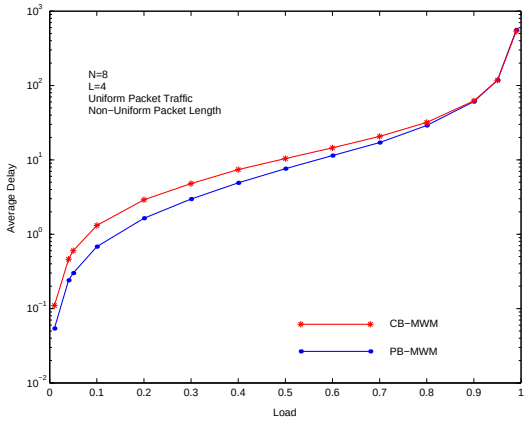


Figure 8: Average delay vs. input traffic load (uniform traffic, non-uniform packet length)

$$P(l) = \begin{cases} 0.3 \text{ for } l = 1, 2, 3 \\ 0.1 \text{ for } l = 4 \end{cases}$$

We can see that for this specific distribution the crossing point of the two curves (PB-MWM and CB-MWM) goes much higher to about 0.95. So PB-MWM outperforms CB-MWM for a wider range of traffic loads.

# 5 Conclusion

In this paper we looked at the packet-based scheduling algorithms. We introduced a general method for converting a cell-based algorithm to a packet-based one. Then, after classifying all packet-based algorithms into waiting and non-waiting categories, with the help of a counter-example we showed that no non-waiting packet-based algorithm can be stable under general admissible traffic. Furthermore, we proved that with i.i.d. traffic assumption, there exist some stable packet-based algorithms. Mainly we proved that PB-MWM is stable for i.i.d. traffic.

Finally, based on some simulation results we concluded that PB-MWM and CB-MWM have approximately similar delay performance. We observe that PB-MWM has better average delay for low traffic loads while CB-MWM is better for higher traffic rates.

We showed that no non-waiting algorithm can be generally stable, so if there exist any generally stable packet-based algorithm it must be a waiting algorithm. An open problem that can be further studied is finding such an algorithm or proving that it does not exist.

# References

[1] Nick McKeown. iSLIP: A scheduling algorithm for input-queued switches, April 1999.

[2] N. McKewon, V. Anantharam, and J. Walrand. Achieving 100% throughput in an input-queued switch. In *Proceedungs of IEEE Info-com 96*, volume 1, pages 296–302, March 1996.

[3] Jim Dai and Balaji Prabhakar. The throughput of data switches with and without speedup. In *Proceedings of the 2000 IEEE Computer and Communications Societies Conference on Computer Communications (INFOCOM-00)*, pages 556–574, Los Alamitos, March 26–30 2000. IEEE.

[4] Balaji Prabhakar and Nick McKeown. On the speedup required for combined input and output queued switching.

Technical Report CSL-TR-97-738, Stanford University, Computer Systems Laboratory, November 1997.

[5] M.A. Marson, A. Bianco, P. Giacone, E. Leonardi, and F. Neri. Scheduling in input-queued cell-based packet switches. In *Proceedings of the Global Telecommunications Conference, GLOBECOM 99*, pages 1227–1235, 1999.

[6] D. Manjunath and Biplab Sikdar. Variable length packet switches: Delay analysis of crossbar switches under poisson and self similar traffic. In *INFOCOM (2)*, pages 1055–1064, 2000.