MNCM a new class of efficient scheduling algorithms for input-buffered switches with no speedup

Vahid Tabatabaee
Department of Electrical and Computer Engineering
and Institute for Systems Research
University of Maryland at College Park
Email: Vahid@glue.umd.edu

Leandros Tassiulas
Department of Electrical and Computer Engineering
and Institute for Systems Research
University of Maryland at College Park
Email: Leandros@glue.umd.edu

***Abstract*— In this paper, we use fluid model techniques to establish some new results for the throughput of input-buffered switches. In particular, we introduce a new class of deterministic maximal size matching algorithms that achieves 100% throughput. Dai and Prabhakar [3] has shown that any maximal size matching algorithm with speedup of 2 achieves 100% throughput. We introduce a class of maximal size matching algorithms that we call them maximum node containing matching (MNCM) algorithms, and prove that they have 100% throughput with no speedup. We also introduce a new weighted matching algorithm, maximum first matching (MFM) with complexity $O(N^{2.5})$ that belongs to MNCM. MFM, to the best of our knowledge, is the lowest complexity deterministic algorithm that delivers 100% throughput. The only assumption on the input traffics is that they satisfy the strong law of large numbers. Besides throughput, average delay is the other key performance metric for the input-buffered schedulers. We use simulation results to compare and study the delay performance of MFM. The simulation results demonstrate promising delay performance for MFM.**

I. Introduction

Due to the progress in optical transmission technology and increase in Internet traffic, very fast switching technology are necessary for internet core and edge switches and routers. Among different switch fabric architectures, input-buffered switches are one of the most popular architectures for the high-speed data networks.

There are three or four major elements in an input-buffered switch fabric architecture. The first element is the input buffer that is used to buffer the incoming cells or packets. The second element is the switching block which is a cross-bar that connects input ports to the output ports. Note that at any time every input can be connected to only one output port and vice versa. The third element is the scheduler that determines which input port to get connected to which output port and configures the cross-bar accordingly. The fourth element is output buffers. Buffering at the output ports is only required if switch fabric has speedup and works at higher rate than the input and output lines.

One of the main reason behind the popularity of the input buffered architecture is that it has the least memory speed requirements. This is specially true when input-buffered switches have no speed-up, because in this case the access rate to cross-bar and memories is equal to the line rate. If we use an input-buffered architecture with speed-up k, then the switch fabric memories and cross-bars should work k times faster than the line rate. In the extreme case, an output-buffered switch is similar to an input-buffered switch with speed-up of N, where N is number of switch ports.

The first challenge of input-buffered switches is their throughput performance. It is a well known fact that due to head of line (HOL) blocking, throughput of input buffered switches for i.i.d Bernoulli arrival pattern is limited to 58.6% [7]. Virtual output queueing (VOQ) can eliminate this problem by maintaining a separate queue for each output in every input [1]. In fact, it is shown that by using suitable scheduling (matching) algorithm the input-buffered switches with VOQs can achieve 100% throughput [12], [8], [9], [3]. The main challenge is to develop and design low complexity scheduling algorithms that can achieve 100% or at least reasonably high throughput.

Stability and throughput of input-buffered switches is a well studied problem. In papers [12], [8] it is proved that maximum weighted matching (MWM) algorithm can achieve 100% throughput. In [8] number of backlogged packets and maximum delay of waiting packets in each VOQ are considered as two potential weight functions. In another work [9], Mekkittikul and Mckeown considered the case where weights are associated to the ports rather than links and showed that the proposed algorithm, longest port first (LPF), achieves 100% throughput. Complexity of LPF is also $O(N^3)$, even though for practical purpose it seems to be more favorable than MWM [10]. Stability of these algorithms are all proven under the assumption of i.i.d. arrivals.

Tassiulas [13] has also introduced a class of randomized iterative algorithms that achieve 100% throughput for i.i.d. arrivals. The complexity of the proposed algorithm is $O(N^2)$, but it is straightforward to introduce an $O(N)$ algorithm in this class too. In [5] modifications to the original algorithm are proposed to improve the performance. One of the problems with randomized scheduling algorithms is their poor and non-

deterministic delay performance. More research is needed to overcome these problems before randomized algorithms become mature enough for consideration in practical systems.

More Recently Dai and Prabhakar [3] have used the fluid model techniques to prove that the maximum link weighted matching achieves 100% throughput for a very general set of input traffic patterns. The only assumption on the input traffic is that it satisfies the strong law of large numbers and it does not over-subscribe any input or output port. In the same work, they have also proved that any maximal matching algorithm with speed up of 2 achieves 100% throughput. This is a very interesting result, because maximal matching algorithms are in general less complex than maximum size and maximum weighted matching algorithms, and therefore are more appropriate for implementation in high speed switches.

In this paper, we extend some of the results of [3], by introducing maximum node containing matching (MNCM) algorithms. MNCM is a new class of maximal matching scheduling algorithms, that achieves 100% throughput with no speedup. The norm function that is commonly used for stability analysis is norm 2 ($\|\cdot\|_2$). In this paper, we use norm infinity ($\|\cdot\|_\infty$) instead of it, and focus on matching algorithms that function based on that. We prove that these matching schemes achieve 100% throughput too. We also introduce maximum first matching (MFM) algorithm that is in MNCM class, and therefore has 100% throughput. MFM employs maximum size matching algorithms with $O(N^{2.5})$ complexity rather than maximum weighted matching algorithms with $O(N^3)$ complexity.

We also introduce another maximal deterministic matching algorithm with $O(N^2)$ complexity, the maximal sorted matching (MSM). MSM is basically a trivial generalization of the iLPF algorithm introduced in [9]. MSM is not in MNCM class, but for practical applications, we think that it performs similar to MFM. Even though we were not able to prove that MSM has 100% throughput, due to its similarity to MFM, we think that for all practical purposes it achieves the 100% throughput. This conjecture is in accordance with our simulation results, where delay performance of MSM matches performance of MFM. Since we have used the fluid model technique, the stability results are under very general conditions for arrival process. The only assumption on the input traffic is that it satisfies the strong law of large numbers. Recall that the results of [9] are for i.i.d. arrivals. Therefore, our results are the first stability results for port weighted matching algorithms under general arrival patterns.

The rest of the paper is organized as follow, in section 2, we introduce our notation and model. In section 3, we review the link based model of the switch, and extend the link based fluid model proposed in [3] to a port based model. In section 4, we prove the main result of the paper, which is the stability of MNCM class of matching. In section 5, we present MFM, and MSM algorithms and elaborate on their complexity. We conclude paper with some simulation results, that demonstrate the delay performance of proposed matching algorithms.

II. Model and Definition

We consider input queued switches that serve fixed size packets (cells). Each input and output has the capacity of serving 1 cell per unit time. Since queues exist only at the input ports, the latter assumption implies that traffic of at most 1 cell per unit time can be transferred from the input ports to a given output port. To avoid HOL blocking, we consider that the buffer at an input is partitioned into N virtual output queues. The scheduling policy is basically a matching algorithm m that based on the state of the switch selects a matching between the inputs and outputs in every time slot. If input i is matched to output j, and the corresponding VOQ is not empty, a packet is sent form input i to output j. A matching can be represented by a permutation matrix π. Input ports are represented by the rows and output ports by the columns of this matrix, therefore input i is matched to output j if and only if $\pi_{ij} = 1$.

We assume that the cells arrive at the switch at the beginning of a time slot, and they depart the switch at the end of a time slot. A packet that has arrived at the beginning of time slot n can be scheduled at the same time slot and depart the switch at the end of time slot n. Let $A_{ij}(n)$ be the number of packets that arrived at input i and are destined for output j up to time n. We assume that there are no arrivals before time 0, i.e., $A_{ij}(0) = 0$. The arrival processes $\{A_{ij}(.), i, j = 1, \ldots, N\}$ satisfy strong law of large numbers, that is with probability one,

$$\lim_{n\to\infty} \frac{A_{ij}(n)}{n} = \lambda_{ij} \quad i, j = 1, \ldots, N. \tag{1}$$

λ_{ij} is the arrival rate of cells destined from input i to output j. Similarly, we show the number of departed cells up to time n, from input i to output j with $D_{ij}(n)$. We consider the following definition for stability.

Definition 1: A switch operating under a scheduling algorithm is rate stable if, with probability one

$$\lim_{n\to\infty} \frac{D_{ij}(n)}{n} = \lambda_{ij} \quad i, j = 1, \ldots, N, \tag{2}$$

for any arrival process that satisfies (1).

Definition 2: A scheduling algorithm is efficient if (2) holds for any arrival process satisfying the feasibility conditions,

$$\begin{aligned} \sum_{i=1}^{N} \lambda_{ij} \leq 1, \\ \sum_{j=1}^{N} \lambda_{ij} \leq 1. \end{aligned} \tag{3}$$

Let $Z_{ij}(n)$ be the number of backlogged packets in VOQ_{ij} at time n, hence

$$Z_{ij}(n) = A_{ij}(n) - D_{ij}(n). \tag{4}$$

We will introduce and analyze some scheduling algorithms that work based on the port level parameters. Port level parameters are defined to be aggregate (summation) of their corresponding link level parameters. We number the ports of the switch from 1 to $2N$ where index sets $\{1, \ldots, N\}$, and $\{N+1, \ldots, 2N\}$ corresponds to input and output ports respectively. Let $(i, j), \quad i, j \in \{1, \ldots, N\}$, indicates one of

the switch links and $k \in \{1, \ldots, 2N\}$ one of the switch ports. We say link (i,j) goes to port k and show it as $(i,j) \to k$, if k and either i or j are associated to the same physical port, i.e. either $k = i$ or $k = j + N$. We define the port arrival process $I(n) = \{I_k(n), k = 1, \ldots, 2N\}$, departure process $E(n) = \{E_k(n), k = 1, \ldots, 2N\}$, the backlogged process $B(n) = \{B_k(n), k = 1, \ldots, 2N\}$, and the port arrival rate $r = \{r_k, k = 1, \ldots, 2N\}$,

$$\begin{aligned} I_k(n) &= \sum_{(i,j):(i,j)\to k} A_{ij}(n), \\ E_k(n) &= \sum_{(i,j):(i,j)\to k} D_{ij}(n), \\ B_k(n) &= \sum_{(i,j):(i,j)\to k} Z_{ij}(n), \\ r_k &= \sum_{(i,j):(i,j)\to k} \lambda_{ij}. \end{aligned} \tag{5}$$

In the next section, we provide the equations that govern evolution of the port level parameters.

III. Link and Port Fluid Models

The fluid model for the links of an input buffered switch is given in [3]. We first review those relations and then extend them to derive the port based fluid model. Consider an input-buffered switch that employs scheduling algorithm m. Suppose that $T_\pi^m(n)$ be the total time that permutation matrix π is used up to time n. The following equations describe the link level discrete dynamic of the switch, for $n \geq 0$, and $i, j = 1, \ldots, N$,

$$\begin{aligned} &Z_{ij}(n) = Z_{ij}(0) + A_{ij}(n) - D_{ij}(n), \\ &D_{ij}(n) = \sum_{\pi\in\Pi} \sum_{l=1}^{n} \pi_{ij} 1_{\{Z_{ij}(l)>0\}} (T_\pi^m(l) - T_\pi^m(l-1)), \\ &\sum_{\pi\in\Pi} T_\pi^m(n) = n, \end{aligned} \tag{6}$$

where Π is the set of all $N \times N$ permutation matrices. The first equation describes the basic relation between arrival, departure and backlogged process. The second equation counts the number of total departures from input i to output j by counting number of times that a permutation matrix with a one in (i,j) position is used, while there were a backlogged packet in the corresponding VOQ.

The port dynamics of a switch can be derived from link dynamics. Every port $k = 1, \ldots 2N$, dynamic is basically summation of its corresponding link dynamics,

$$\begin{aligned} B_k(n) &= B_k(0) + I_k(n) - E_k(n), \\ E_k(n) &= \sum_{\pi\in\Pi} \sum_{l=1}^{n} \sum_{(i,j):(i,j)\to k}^{N} \\ &\qquad \left(\pi_{ij} 1_{\{Z_{ij}(l)>0\}} (T_\pi^m(l) - T_\pi^m(l-1))\right), \\ \sum_{\pi\in\Pi} T_\pi^m(n) &= n. \end{aligned} \tag{7}$$

Now we describe a deterministic continuous fluid model of a switch operating under some matching algorithm m. The link level fluid model that is used in [3] is,

$$\begin{aligned} &Z_{ij}(t) = Z_{ij}(0) + \lambda_{ij} t - D_{ij}(t) \geq 0, \\ &\dot{D}_{ij}(t) = \sum_{\pi\in\Pi} \pi_{ij} \dot{T}_\pi^m(t) \geq 0, \quad \texttt{if} \quad Z_{ij}(t) \geq 0, \\ &\sum_{\pi\in\Pi} T_\pi^m(t) = t. \end{aligned} \tag{8}$$

We can derive the port level fluid model of the switch from relation 8 and 5, or directly from 7.

$$\begin{aligned} B_k(t) &= B_k(0) + r_k\ t - E_k(t) \geq 0, \\ \dot{E_k}(t) &= \sum_{\pi\in\Pi} \sum_{(i,j):z_{ij}(t)>0 \& (i,j)\to k} \pi_{ij} \dot{T}_\pi^m(t) \geq 0, \\ \sum_{\pi\in\Pi} T_\pi^m(t) &= t. \end{aligned} \tag{9}$$

Note that (9) is not an independent complete set of equations that describes the dynamic behavior of the port level fluid model for the switch, since in the second equation we still use link level parameters $(z_{ij}(t))$ of the switch.

The matching algorithm that is used for scheduling provides additional fluid model equations. For instance, if we use the conventional maximum weighted matching algorithm and consider number of backlogged cells of every link, $Z_{ij}(t)$ as its weight, there will be one additional fluid equation for every link (i,j) [3],

$$\dot{T}_\pi^m(t) = 0 \quad if \quad \langle \pi, Z(t) \rangle < \langle \pi', Z(t) \rangle \ \text{ for some } \pi' \in \Pi\ . \tag{10}$$

In other words, at time t, under the maximum weight matching algorithm, a matching π that has less weight than another matching π' is not employed.

In this paper, we consider weighed matching algorithms that function based on the port weight rather than link weights. Weight of a port is total weight of all links connected to it. LPF [9] is one of these algorithms, where we consider the backlogged traffic in every port as the weight of that node. Maximum node matching (MNM) [14] is another example that weights are amount of service that scheduler owes every link, according to the reserved rate of that port. The only difference between the MNM and LPF is in the weight function that is

used for links, but they have the same complexity. It is proved in [14] that the obtained matching under MNM is a min-max matching, and it has the maximum lexicographic ordering. In other words, it is impossible to add a new port to the matching, without removing a higher weight node from the matching. This is also true for LPF for the corresponding weight vectors. This property can be used to write an additional fluid equation for LPF, however we consider a generalized class of matching algorithms that includes LPF, and derive some interesting result for this general class of matching algorithms. The significance of MNCM class becomes more clear later, when we introduce a deterministic matching algorithm in MNCM that has lower complexity, and has a very good delay performance. First we need to define MNCM class:

Definition 3: A maximal size matching algorithm m belongs to MNCM class if and only if m contains all nodes with maximum weight.

For now, we assume that MNCM is a non-empty class, and for any combination of link weight values there exists at least one MNCM matching. Under this assumption, LPF turns out to be MNCM. Recall that LPF is a min-max matching algorithm. Therefore, if there is any matching that contains all nodes with maximum weight, LPF should contain all maximum weight ports too, otherwise it is not a min-max matching. In the next section, we prove that any MNCM scheduler is efficient, and this immediately proves that LPF is efficient. However, there are some less complex algorithms in MNCM class too. Basically, a matching is in MNCM that at least covers those nodes with maximum weight, but we do not care about the rest of the nodes and wether they are included or not.

Recall that the matching algorithm that is used for scheduling algorithm results in additional fluid model equations. For any matching in MNCM the following additional fluid equation holds. For any port k with maximum weight at time t we have,

$$\sum_{\pi\in\Pi}\Big(\sum_{(i,j):z_{ij}(t)>0\&(i,j)\to k}\pi_{ij}\Big)\dot{T}^m_\pi(t) \quad = 1 \qquad \text{if } B_k(t) \text{ is maximum} \tag{11}$$

Equation (11) says that all ports with maximum weight are fully served under a MNCM policy.

Before ending this section we need to define function $f(B(t))$. Function $f(B(t))$ is defined from $R^{2N}\to R$ as,

$$f(B(t)) = \max\{B_1(t),\ldots,B_{2N}(t)\} \tag{12}$$

This function plays a role similar to the Lyapanov function in the stability proofs. Under a fluid model, $B(t)$ is differentiable, and consequently, $f(B(t))$ is continuous but not necessarily differentiable. However, due to the differentiability of $B(t)$, we can define the right derivative of $f(.)$ as follow,

$$\frac{df(B(t))}{dt^+} = \lim_{\delta\to 0^+}\frac{f(B(t+\delta)-f((B(t))}{\delta} \tag{13}$$

We use $f(B(t))$ and its right derivative in the course of stability proofs.

IV. Stability Results

Our main objective here is to prove the following theorem,

Theorem 1: A switch operating under an MNCM matching algorithm is efficient.

To prove theorem 1, we will use the following theorem that is proved in [3].

Theorem 2: A switch operating under a matching algorithm is rate stable if the corresponding fluid model is weakly stable.

A fluid model is weakly stable if for every fluid model solution (E,T,B) with $B(0)=0$, $B(t)=0$ for all $t\geq 0$.

Before proceeding further, we need to elaborate on the properties of function $f(B(t))$. Clearly for every $t_0\geq 0$ there is always a subset of indices $Q(t_0)$ such that for every $k\in Q(t_0)$, $f(B(t_0))=B_k(t_0)$, i.e., $B_k(t_0)$ is the maximum entry of the vector $B(t_0)$. We say that $B(t_0)$ is represented by $Q(t_0)$ at time t_0. Moreover, due to continuity properties of $B(t)$ under the fluid model, for every $t_0\geq 0$, there exists some $\delta>0$ such that for all $t\in[t_0,t_0+\delta[$ there is always one common index $q(t_0,t_0+\delta)\in Q(t)$. We say that $B(t)$ is represented by $q(t_0,t_0+\delta)$ in $[t_0,t_0+\delta[$ interval, i.e. $B_{q(t_0,t_0+\delta)}$ is maximum entry of $B(t)$ in the interval $[t_0,t_0+\delta[$. We can always partition the time axis into disjoint intervals $[0,\Delta_1[,[\Delta_1,\Delta_1+\Delta_2[,\cdots$ such that each interval is represented by a common index q_i $i=1,2,\cdots$. We say that the sequence $\{\Delta_i\}_{i=1}^{\infty}$ is a D-partition, where D stands for differentiable.

The point is that in every sub-interval of a D-partition, $f(B(t))$ is equal to one entry of $B(t)$, and hence it is differentiable with respect to t. Since these intervals are open in right, for any such interval i that is represented by q_i we can write,

$$\Big(\frac{df(B(t))}{dt^+}\Big)_{t\in\ i-th\ interval} = \dot{B}_{q_i}(t_0). \tag{14}$$

Now we are in a position to prove the theorem,

Proof of theorem 1: Let (E,T,B) be a solution to equations (9), (11) with $B(0)=0$. In order to prove that $B(t)$ remains zero for $t>0$, it suffices to prove that $f(B(t))$ remains zero. To that end we show that,

$$\frac{df(B(t))}{dt^+}\leq 0\ \forall t>0.$$

Suppose that the sequence $\{\Delta_i\}_{i=1}^{\infty}$ is a D-partition for solution (E,T,B) with $B(0)=0$, and q_i represents $f(B(t))$ in

i-th interval. Let $t_0 \geq 0$ resides in the i-th interval. We have,

$$
\begin{aligned}
(\frac{df(B(t))}{dt^+})_{t=t_0} &= \dot{B}_{q_i}(t_0) \\
&= r_{q_i} - \dot{E}_{q_i}(t_0) \\
&= r_{q_i} - \sum_{\pi \in \Pi} \\
&((\sum_{(p,j):Z_{pj}(t_0)>0 \& (p,j)\to q_i} \pi_{pj})\, \dot{T}^{m}_{\pi}(t_0)) \\
&= r_{q_i} - 1 \\
&\leq 0
\end{aligned}
$$

The second and third relations are direct application of fluid model equations given in (9). The fourth equation is due to property of MNCM given in (11), and the fact that B_{q_i} is maximum. The last equation is from feasibility condition given in (3). Therefore, we have proved that $Z(t) = 0$ and fluid model is weakly stable. Using theorem 2, we can conclude that the MNCM policies are efficient.

$\diamond$

The novelty of our approach is to consider the maximum norm in definition of function $f(\cdot)$. In the previous approaches usually a second norm function was used in definition of $f(\cdot)$. In the next section, we introduce a matching algorithm that is in MNCM and has a lower complexity than all previous suggested deterministic algorithms.

V. Maximum First Matching (MFM)

In the previous section we proved that MNCM algorithms are efficient. In the proof, we implicitly assumed that MNCM algorithms always exist. We will prove that this is true and then introduce MFM as a low complexity member of this class.

Lemma 1: Consider an $N \times N$ switch, and suppose that $B_i(t)$ $i = 1, \cdots, 2N$ is the number of buffered packets corresponding to port i in the switch. There always exist a matching $m \in MNCM$.

Proof of lemma 1: We consider different cases according to the number of nodes with maximum weight. The case where there are at most one node in input side and one in output side with maximum weight is trivial. Suppose that there are more than one node in the input side with maximum weight and they can not be all matched to the output nodes. We show that this can not happen. Using Hall's theorem [2], this means that there is a subset S of the maximum weight nodes with cardinality $k > 1$ that has less than k neighbors in the output side. Let $N(S)$ be the neighbor set of S,

$$|N(S)| < |S| = k. \tag{15}$$

On the other hand, total weight of $N(S)$ nodes is at least equal to the total weight of S, because all links that are connected to S are also connected to $N(S)$. Let $W(S)$ be the total weight of nodes in set S, and $W(N(S))$ total weight of nodes in $N(S)$.

$$W(N(S)) > W(S) = k\, Z_{\max} \tag{16}$$

From (15) and (16) we can say that there should be at least one node in the neighbor set that its weight is at least $k/(k-1)Z_{\max}$. This contradicts with the assumption that $Z_{\max}$ is the maximum weight.

$\diamond$

Note that as long as the node weight is defined as summation of link weights the proof is valid, weight function does not necessarily need to be number of backlogged cells. The LPF algorithm that is introduced by McKeown [8] is an example of MNCM algorithms. LPF works on non-matched nodes one by one starting with the node that has highest weight. To find the matching, for each one of the nodes we need to search all N^2 links of the bipartite graph. Since there are $2N$ nodes, the complexity of LPF turns out to be $O(N^3)$. Basically this algorithm is a modification of the Edmonds Karp max-flow algorithm [4] and its complexity is the same as that.

The only introduced matching algorithm with $O(N^{2.5})$ complexity is Hopcroft and Karp algorithm [6], which is a maximum size matching algorithm. In this algorithm the nodes are introduced into the matching simultaneously; this is not possible in LPF algorithm.

In MFM algorithm, we convert the maximum weighted matching algorithm to a limited number of maximum size matching algorithms. In this way, we can use the Hopcroft and Karp algorithm to obtain a simple maximum size matching in each step,and reduce the complexity of the algorithm. We can do this since our objective is no more to find the maximum weighted matching, but to have a matching that contains all node with maximum weight. Since MFM contains all nodes with maximum weight it is MNCM and is therefore efficient. The details of the algorithm is as follow (complexity of each step is given at the end of each step),

Algorithm 1: (MFM):

1) Sort all input and output nodes according to their weight ($O(N \log N)$).
2) Find a matching M_1 that contain all input nodes with max weight ($O(N^{2.5})$).
3) Find a matching $M2$ that contain all output nodes with max. weight ($O(N^{2.5})$).
4) Combine M_1 and M_2 to get an inclusive matching $M_1 \cup M_2$ ($O(N)$).
5) Perform a simple sorted maximal matching algorithm on the rest of nodes not in the matching ($O(N^2)$).
6) The combination of matchings of step 4 and 5 is MFM matching.

$\diamond$

In step 1, all nodes are sorted according to their weights. Since there are $2N$ nodes the complexity of this step is

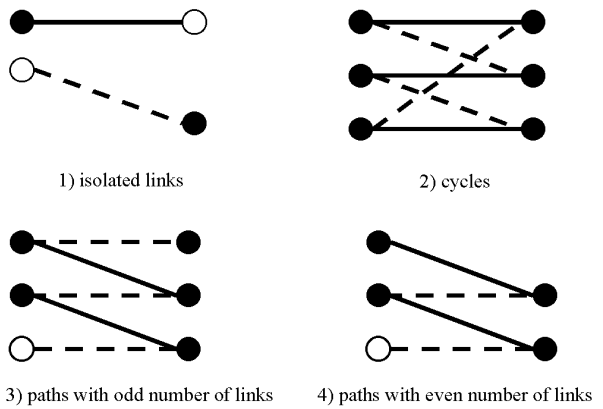

Fig. 1. Four possible forms of sub-graphs in a combined matching. Links of M_1 are shown with solid lines and M_2 with dashed lines. Nodes with maximum weight are shown as black nodes.

$O(N \log N)$. In step 2 we consider input nodes with maximum weight together with all output nodes and find a maximum size matching, M_1, in the corresponding graph. The complexity of this step is $O(N^{2.5})$ and from lemma 1 we know that it would cover all input nodes with maximum weight. Step 3 is similar to step 2, but here we find a matching, M_3 that covers all output nodes with maximum weight. In step 4, we combine the two matchings to find a matching that cover both input and output nodes with maximum weight.

It is useful to elaborate more on step 4. Consider the bipartite graph that contains only those links that are in M_1 and M_2. Our objective is to find a matching that contain all of the nodes with maximum weight from these two matchings. The maximum degree of a node in the combined graph is 2, since there are at most two links connected every node (one from M_1 and one from M_2). Therefore this bipartite graph can be divided into disjoint sub-graphs. For each subgraph we have to obtain an optimal matching. Subgraphs can be classified into four classes and the process of obtaining a matching is different for each class. In general, we have to select a group of links, belonging to either M_1 or M_2, that construct a matching in the sub-graph as follow (Fig.1)

1) **Single link subgraph:** If there is an isolated link connecting two nodes that link is included in the matching.
2) **Cycle subgraph:** Since the graph is bipartite, cycles have even number of links. Links alternatively belong to M_1 and M_2. We can select either set of the links for matching, since both cover all nodes in the cycle.
3) **Path subgraph with odd number of links:** Here we have an alternate path, that is similar to an augmented path in matching terminology. Basically one set of links either those belonging to M_1 or M_2 has one more element. That set covers all nodes in the sub-graph, and thus should be selected for matching.
4) **Path subgraph with even number of links:** Without loss of generality, assume such a path that starts from an input node. Obviously, since there are even number of links this path ends also at an input node. One of these nodes belongs to M_1, and therefore has maximum weight the other belongs to M_2 and is not a maximum weight node (because it is not in M_1). If we select those links that belong to M_1, only that node that does not have maximum weight will be secluded, which is not important.

Therefore, we can come up with the following general rule, if a path starts from an input (output) node that has maximum weight include those set of alternating links in the matching that contain that particular node.

To determine the complexity of step 4, notice that in this step we have to search the graph that is obtained from $M_1 \cup M_2$. This graph has $2N$ links,and therefore the search complexity is $O(N)$. The ultimate matching covers all critical nodes with maximum weight.

In step 5, we perform a sequential maximal matching on the nodes that are not in the matching. Starting from the node with maximum weight, we scan all its neighbors starting from the one with maximum weight. If we find a neighbor that is not in the matching we match and include the corresponding link in the matching list. The number of nodes that we have to scan in this step is less than $2N$, and for each one we have to check at most N neighbors. This has a complexity of $O(N^2)$

Even though the complexity of this algorithm is $O(N^{2.5})$, in practice it would be less than that. Practically number of nodes with maximum weight is very limited, and therefore complexity of steps 2,3,4 is very low, and it is step 5 that has the highest complexity and this results in $O(N^2)$ complexity algorithm.

One may think that there should be lower complexity algorithms that are practically efficient and has lower complexity than MFM. Maximal sorted matching (MSM) can be considered as a first attempt toward such an algorithm. MSM is basically very similar to iLPF algorithm that is introduced in [9]. Similar to MFM, MSM also works on a sorted list of the nodes, however it does not treat the nodes with maximum weight separately to ensure that they are contained into the matching. MSM scans all nodes starting with the one with maximum weight and going down the sorted list, and tries to include the scanned (primary) nodes into the matching. At every step it scans all neighbor nodes (starting from the neighbor node with maximum weight) of the primary node until it finds a free (not matched) node or until it scans all neighbors. If it finds a free neighbor, it considers it as the secondary node and match the primary and secondary nodes and add the connecting link to the matching. This is essentially similar to step 5 of MFM and results in an $O(N^2)$ complexity algorithm.

Algorithm 2: (MSM):

1) Sort all input and output nodes according to their weight $O(N \log N)$.

2) Perform a simple sorted maximal matching algorithm on all of the nodes ($O(N^2)$).

$\diamond$

Although we can not prove that MSM achieves 100% throughput, in simulations its performance was identical to MFM. We review some of the simulation results in the next section.

VI. Simulations

In this section we present our simulation results. The main objective is to study delay performance of the proposed scheduling algorithms. For practical purposes it is not sufficient for a scheduling algorithm to have 100% throughput. Therefore, it is essential to compare and study the delay performance of different scheduling algorithms.

We consider a 32×32 input-buffered switch. For each experiment the average throughput, ρ, of the switch is given. A random 32×32 rate matrix is generated such that the aggregate rate of every input (row summation) and output (column summation) is ρ.

The rate matrix is generated iteratively. In each iteration a new flow between two randomly selected input and output nodes is generated. After selecting the input node i, and output node j, of a flow the maximum allowable rate (MAR) for that flow is set to minimum of:

- Maximum flow rate that is set to 0.1 in our experiments.
- Difference between ρ and aggregate rate of input port i. Aggregate rate of i is summation of all elements of rate matrix in row i.
- Difference between ρ and aggregate rate of output port j. Aggregate rate of j is summation of all elements of rate matrix in column j.

Next, a uniform random number between zero and MAR is generated as the rate of that flow and it is added to the (i, j) element in rate matrix. In fact, MAR is generated such that the aggregate rate of each port does not exceed ρ. This procedure is repeated until aggregate rates of all ports become very close to ρ. In this way, we are able to create a non-uniform rate matrix. This matrix is used to generate independent Bernoulli arrival patterns for all of the connections.

We have studied four different systems which are three input-buffered switches with matching algorithms LPF, MFM, MSM , and an output buffered switch. The first three systems were simulated, but for the output-buffered system we modelled it as an M/D/1 system. We have elaborated on the first three, and proved here that LPF and MFM achieve 100% throughput. The output-buffered system is mainly included as a bench mark measure.

The main performance measure that we have considered here is delay versus throughput, and it is plotted for all of the scheduling algorithms in Fig. 2. For each scheduler the simulation is stopped, when the throughput reached 1.

As we expected all of the systems achieve 100% throughput. Delay performance of LPF is slightly better than MFM and MSM, but recall that it is more complex as well. The performance of MFM and MSM is not distinguishable. Although we have not been able to prove the stability of MSM, the simulations reveal that for practical purposes it function as 100% throughput algorithm, with suitable delay performance. Recall that MSM complexity is $O(N^2)$, which is lower than other proposed algorithms.

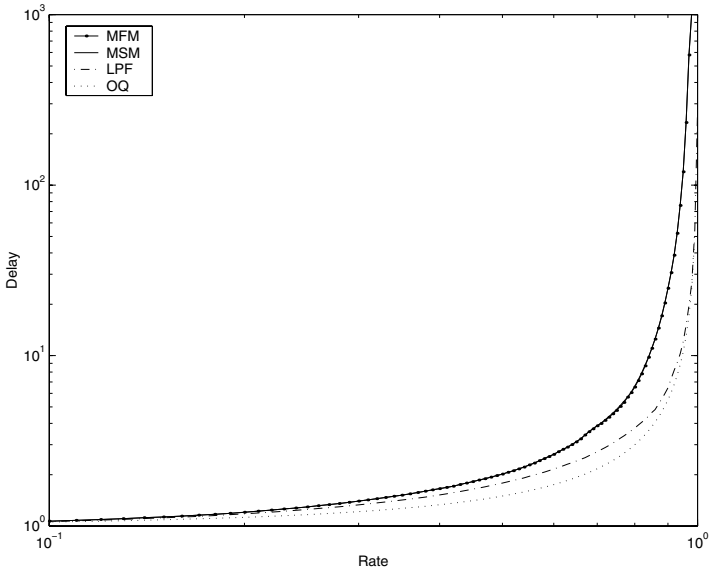

Fig. 2. Average Delay v.s. Throughput for different matching algorithms. The output queueing system is modelled as an M/D/1 system.

VII. Summary

In this paper, a new class of weighted matching algorithms for scheduling in input-buffered switches is introduced. Using the fluid model techniques, we were able to prove that they achieve 100% throughput. Basically, it is shown that to achieve 100% throughput it suffices to include only nodes with maximum weight in the matching. This fact can lead us to development of less complex and more efficient scheduling algorithms for input-buffered switches. Two particular matching algorithms, MSM and MFM that can be considered for practical systems, are also introduced. MFM is proven to achieve 100% throughput. In addition, the simulation results for MFM and MSM have been very similar. Throughout the paper we have used number of backlogged cells as the weight of a port, however there are other proposed weight functions in the literature that for example can provide rate provisioning [14]. The proofs in the paper can be generalized to prove that under similar conditions to what stated in the paper, if we use rate provisioning weight functions instead of number of backlogged packets, then the weight of all ports remain bounded. This means that if the reserved rate of all ports is less than the total capacity of ports, the obtained matching algorithms would be able to sustain the port to port conformed rates. We are currently studying the behavior of the proposed algorithms with alternative weight functions that can provide rate provisioning.

REFERENCES

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

[2] T.H. Cormen, C.E. Leiserson, R.L. Rivest, "Introduction to algorithms", *The MIT Press* 1990.

[3] J.G. Dai, B. Prabhakar, "The throughput of data switches with and without speedup", *INFOCOM '00*

[4] J. Edmonds, R.M. Karp, "Theoretical improvements in the algorithmic efficiency for network flow problems", *Journal of the ACM,*v.19, 248-264, 1972

[5] P. Giaccone, B. Prabhakar, D. Shah, "Towards simple, high-performance schedulers for high-aggregate bandwidth switches", *INFOCOM '02*

[6] J.E. Hopcroft, R.M. Karp "An $n^{5/2}$ algorithm for maximum matching in bipartite graphs", *SIAM Journal on Computing,*v.2, no.4, 225-231, 1973

[7] M.J. Karol, M.G. Hluchyj, S.P. Morgan, "Input Versus Output Queueing on a Space-Division Packet Switch", *IEEE Transactions on Communications,*v.35, 1347-1356, 1987

[8] N. McKeown, V. Anantharam, J. Warland, " Achieving 100% Throughput in an Input-Queued Switch", *INFOCOM '96*, 296-302.

[9] A. Mekkittikul, N. McKeown, " A Practical Scheduling Algorithm to Achieve 100% Throughput in Input-Queued Switches", *INFOCOM '98*, 792-799.

[10] A. Mekkittikul, N. McKeown, " Scheduling VOQ switches under non-uniform traffic", *CSL Technical Report,* CSL-TR-97-747, Stanford University, 1997.

[11] N. McKeown, " iSLIP: A scheduling algorithm for Input-Queued Switches *IEEE Transaction on Networking",* v.7, 188-201, April 1999.

[12] L. Tassiulas, A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks", *IEEE Trans. on Automatic Control*, v.37, n.12, Dec. 1992, pp. 1936-1948.

[13] L. Tassiulas,"Linear Complexity Algorithms for Maximum Throughput in Radio Networks and Input Queued Switches", *Infocom98*, 533-539.

[14] V. Tabatabaee, L. Georgiadis, L. Tassiulas, " QoS Provisioning and Tracking Fluid Policies in Input Queueing Switches", *IEEE Transaction on Networking*, v.9, no.5, pp. 605-617, Oct. 2001.