



MFM Forecast Algorithm for Input-Queued Switch Under Non-Consistent Traffic

V. Rajesh Kumar

Assistant Professor, Department of CSE,
Alpha College of Engineering and Technology, Pondicherry.
India

V. Vimala Devi

Assistant Professor, Department of CSE
P.S.R.R.C.E.W. Sivakasi, Tamil Nadu
India

Abstract- In this paper, the Maximal Node Containing Matching (MNCM), is a latest set of matching algorithms that achieve 100% throughput with no speedup. The MNCM policies only need to include ports whose weight (backlog) are above a threshold in the matching rather than finding a matching with maximum total weight. This simplified requirement enables to introduce a new matching algorithm, Maximal First Matching (MFM), with $O(N^{2.5})$ complexity. It shows that MFM is a low complexity algorithm with good delay performance. It also compares the performance of MFM with other matching scheduling algorithms (LQF and MSM), and the simulation results illustrate that MFM provides the best delay-throughput performance.

Keywords- Maximal Node Containing Matching, Maximal First Matching, threshold, no speedup

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 it 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 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 tight functions. In another work [9], Mekittikul 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, it extends 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. In this paper, it uses norm infinity instead of it, and focuses on matching algorithms that function based on that. It proves that these matching schemes achieve 100% throughput too. It also introduces 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. The rest of the paper is organized as follows: in section 2, it introduces our switch model. In section 3, it reviews the existing algorithms LQF and MSM. In section 4, it introduces our proposed algorithm MFM. In section 5, it proves the main result of the paper. It concludes the paper with some simulation results, that demonstrate the delay performance of the proposed matching algorithm.

II. System Model

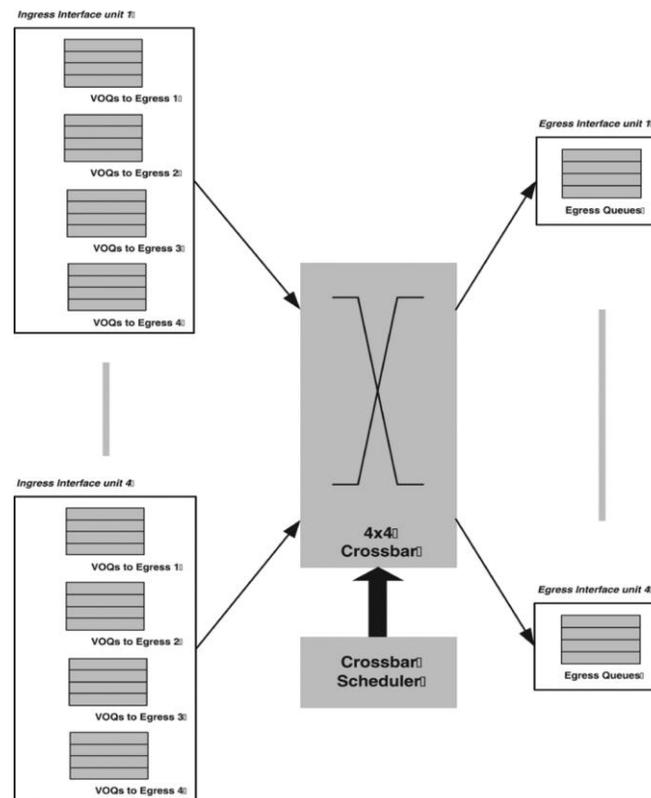


Figure 1: $N \times N$ input-queued switch fabric architecture.

III. Existing Algorithms

Algorithm 1: (LQF)

scheduling algorithm for an input-queued switch consists of two parts. First it must choose which input is matched to which output. (This decision is made by LQF.) Once a port-pair is chosen it must then decide which packet to transmit. Longest-Queue-First (LQF) protocol the number of packets waiting to traverse the switch remains bounded as long as no input port or output port is overloaded. The LQF protocol calculates a maximum-weight bipartite matching at each time step, where the weight on edge $(i;j)$ equals the number of packets that are waiting to travel from input port i to output port j . However, one feature of this result and other literature on input-queued switches is that it focuses on a single switch in isolation

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 ($O(N^2)$).

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

IV. Proposed Algorithm

MAXIMUM FIRST MATCHING (MFM)

In the previous section it proved that MNCM algorithms are efficient. In the proof, it implicitly assumed that MNCM algorithms always exist. It will prove that this is true and then introduce MFM as a low complexity member of this class. 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 it 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, it convert the maximum weighted matching algorithm to a limited number of maximum size matching algorithms. In this way, it can use the Hopcroft and Karp algorithm to achieve a plain upper limit matching in each stride, and decrease the convolution of the algorithm. It 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 upper limit weight it is MNCM and thus it is efficient. The information of the algorithm is as go behind (complexity of each step is given at the end of each step),

Algorithm (MFM):

- 1) Sort all key in and amount produced nodes according to their weight ($O(N \log N)$).
- 2) Find a matching $M1$ 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 $M1$ and $M2$ to get an inclusive matching $M1 \cup M2$ ($O(N)$).
- 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.

In step 1, all nodes are sorted according to their weights. Since there are $2N$ nodes the complexity of this step is of $M1$ are shown with solid lines and $M2$ with dashed lines. Nodes with maximum weight are shown as black nodes. $O(N \log N)$:

In step 2 it consider input nodes with maximum weight together with all output nodes and find a maximum size matching, $M1$, in the corresponding graph. The complexity of this step is $O(N^2.5)$ and from lemma 1 it know that it would cover all input nodes with maximum weight.

Step 3 is similar to step 2, but here it find a matching, $M3$ that covers all output nodes with maximum weight. In step 4, it 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 $M1$ and $M2$. 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 $M1$ and one from $M2$). Therefore this bipartite graph can be divided into disjoint sub-graphs. For each subgraph it 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, it have to select a group of links, belonging to either $M1$ or $M2$, 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 $M1$ and $M2$. It 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 it have an alternate path, that is similar to an augmented path in matching terminology. Basically one set of links either those belonging to $M1$ or $M2$ 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 $M1$; and therefore has maximum weight the other belongs to $M2$ and is not a maximum weight node (because it is not in $M1$). If it select those links that belong to $M1$, only that node that does not have maximum weight will be secluded, which is not important. Therefore, it 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 it have to search the graph that is obtained from $M1 \cup M2$: 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, it perform a sequential maximal matching on the nodes even which will be not in the matching. Starting from the node with maximum weight, it scan all its neighbors starting from the one with maximum weight. If it find a neighbor that is not in the matching it match and include the corresponding link in the matching list. The number of nodes that it has to scan in this stride is not as much of than $2N$; and for each one it have to check at most N neighbors. This has a convolution of $O(N^2)$. Even though the convolution of this algorithm is $O(N^2.5)$, in perform it would be a lesser amount of 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 loitr

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.

V. Simulation

In this section it presents a 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.

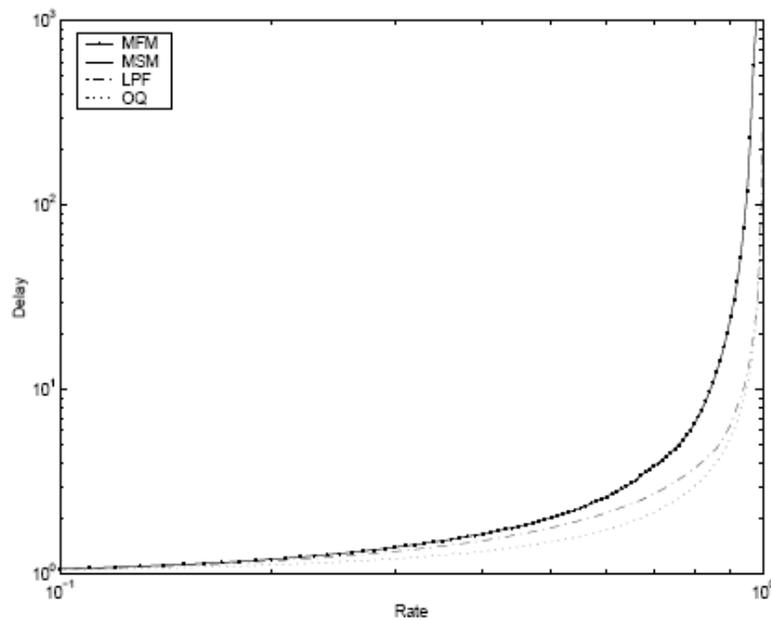


Figure. 2. Average Delay v.s. Throughput for different matching algorithms.

It 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 are simulated, but for the output-buffered system it modelled it as an M/D/1 system. It 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 it 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 it expected all of the systems achieve 100% throughput. Delay performance of MFM is better than LQF and MSM

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. Edmonds, R.M. Karp, "Theoretical improvements in the algorithmic efficiency for network flow problems", Journal of the ACM, v.19, 248-264, 1972
- [4] P. Giaccone, B. Prabhakar, D. Shah, "Towards simple, high-performance schedulers for high- aggregate bandwidth switches", INFOCOM '02
- [5] A. Mekikittikul, N. McKeown, "A Practical Scheduling Algorithm to Achieve 100% Throughput in Input-Queued Switches" INFOCOM '98, 792-799.
- [6] A. Mekikittikul, N. McKeown, "Scheduling VOQ switches under non-uniform traffic", CSL Technical Report, CSL-TR-97-747, Stanford University, 1997.

- [7] N. McKeown, " *iSLIP: A scheduling algorithm for Input-Queued Switches* *IEEE Transaction on Networking*", v.7, 188-201, April 1999.
- [8] 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.
- [9] L. Tassiulas, " *Linear Complexity Algorithms for Maximum Throughput in Radio Networks and Input Queued Switches*", *Infocom98*, 533-539.
- [10] 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.