Randomized scheduling algorithm for virtual output queuing switch at the presence of non-uniform traffic

Ali Ghiasian[1], Majid Jamali[2]
[1]Faculty of Engineering, Shahrekord University, Shahrekord, Iran
[2]Aghigh University, Isfahan, Iran

Article Info

Article history:

Received Nov 13, 2018
Revised Dec 29, 2018
Accepted Jan 20, 2019

Keywords:

Maximum weight matching
Scheduling
Switch
Virtual output queuing

ABSTRACT

Virtual Output Queuing (VOQ) is a well-known queuing discipline in data switch architecture that eliminates Head Of Line (HOL) blocking issue. In VOQ scheme, for each output port, a separate FIFO is maintained by each input port. Consequently, a scheduling algorithm is required to determine the order of service to virtual queues at each time slot. Maximum Weight Matching (MWM) is a well-known scheduling algorithm that achieves the entire throughput region. Despite of outstanding attainable throughput, high complexity of MWM makes it an impractical algorithm for implementation in high-speed switches. To overcome this challenge, a number of randomized algorithms have been proposed in the literature. But they commonly perform poorly when input traffic does not uniformly select output ports. In this paper, we propose two randomized algorithms that outperform the well-known formerly proposed solutions. We exploit a method to keep a parametric number of heavy edges from the last time matching and mix it by randomly generated matching to produce a new schedule. Simulation results confirm the superior performance of the proposed algorithms.

Corresponding Author:

Ali Ghiasian,
Faculty of Engineering,
Shahrekord University, Shahrekord, Iran.
Email: Ghiasian.ali@eng.sku.ac.ir, jmajid62@gmail.com

1. INTRODUCTION

High-speed packet switching is the essential technology of the communication networks. A data switch (also called switch in this paper) is a networking device with *N* input and *N* output ports. It receives packets from input ports, process them, and finally forward them towards the destination device through its appropriately distinguished output ports. When two or more packets arrive at the same time slot from different input ports and are destined to the same output port, congestion occurs within the switch and some packets will have to be queued in buffers for a while. The architecture of switches can be classified by the queuing buffer position. In Input Queuing (IQ) architecture, the buffers are placed in the input ports while in Output Queuing (OQ), the output ports are hosting the buffers. In IQ and OQ, the buffers are dedicated to specific ports while in Shared Buffer (SB) structure, the memory for saving packets are shared among input and output ports. Combinations of the mentioned methods are also possible which is termed as Combined Input Output Queuing (CIOQ). In recent technology trend, single buffers could be implemented at the cross points in cross bar switches [1]. A cross bar is a common single-stage switch fabric consists of a matrix of N^2 cross points.

In an OQ switch, a memory bandwidth equal to N times link rate is required if no packet loss is acceptable. This limits the scalability of the OQ switch[2]. IQ switch suffers from Head Of Line (HOL) blocking issue which limits the throughput region of the switch to 58.6% for Bernoulli packet arrivals with uniformly selected output ports [2]. The mentioned type of traffic is termed uniform traffic throughout the paper. The solution to this problem is to queue packets in different buffers associated to output ports. The new architecture is termed as Virtual Output Queuing (VOQ) as shown in Figure 1.

Associated to Figure 1, a Bi-partite graph is defined as G(V,E) where V is the set of input and output ports and E is referred to the set of edges. To each edge (i,j) a weight Wij is assigned which is equal to the corresponding queue length from port i to port j. The graph of Figure 1 is shown in Figure 2.

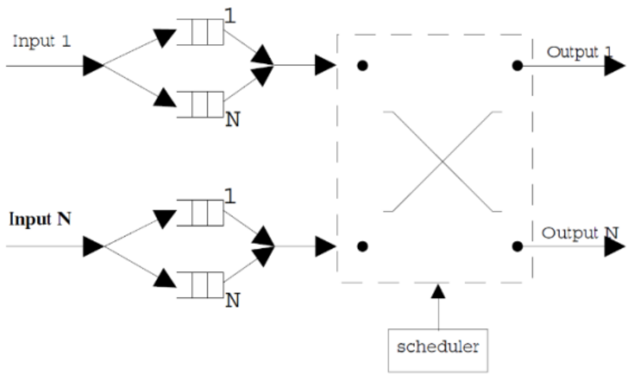

Figure 1. Virtual output queuing switch

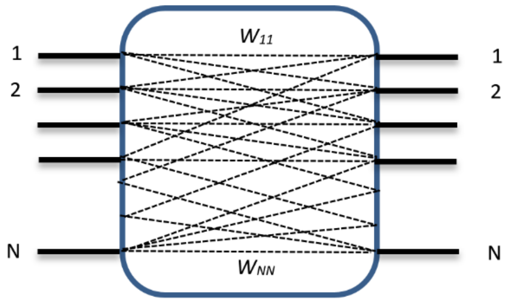

Figure 2. Bi-Partite graph of Figure 1

To achieve high throughput in VOQ architecture, different packets destined to the same port should be scheduled to prohibit collision. A Scheduling algorithm determines the order of service to virtual queues at each time slot. In other words, it defines a *Bi-Partite Matching*, at each time slot in graph G. The scheduling discipline highly affects the delay of packets as well as the switch throughput. *Maximum Weight Matching (MWM)* is a scheduling algorithm that achieves the entire throughput region[3]. In MWM context, a weight is assigned to each queue. The assigned weight can be the length of the queue in terms of bytes, or can be the delay of the first packet in the head of queue. The former is known as Longest Queue First (LQF) and the latter is named Oldest Cell First (OCF) algorithm [3]. LQF and OCF are samples of scheduling algorithms that provably achieve 100% of the throughput region for uniform traffic [3]. These algorithms are termed *throughput optimal*. Not only the queue length, but a broad class of weight functions such as square and cube of queue length give 100% throughput [4]. However, the delay behaviour of these functions are different.

Modified Largest Weighted Delay First (M-LWDF) algorithm is a throughput optimal discipline that is devised for variable channel rate condition [5]. It is a modified version of MWM in which the variable link capacity is taken in to account.

The big challenge of MWM algorithm is its complexity for implementation in high-speed switches. The complexity of MWM is O(N^3) and it requires many control messages to be communicated between input and output ports [6]. To overcome the implementation issue of MWM, a number of scheduling algorithms, such as iSLIP [7] and RPA [8] have been proposed in literature. iSLIP can achieve 100% throughput for uniform traffic. But, when input traffic is not uniform, these algorithms perform poorly in terms of delay and their achievable throughput is less than 100%.

A number of approximation to MWM are proposed in the literature to reduce the implementation complexity in cost of either increased delay or achieving a fraction of throughput region. In [9] a class of approximation algorithms to MWM has been studied which obtain a schedule whose weight differ from MWM weight by at most a sub-linear function of the weight of MWM. While those algorithms in [9] are throughput optimal, the delay bound of the packets is linearly related to the difference of the weight of approximate schedule and MWM. In [10] the average delay under MWM scheduling was proven to be upper bounded by O(N). The proposed solutions towards reducing complexity are commonly increasing the average delay. The trade-off in complexity and delay was shown in [11]. A frame-based scheduling with O(log(N)) delay bound was proposed in [12]. A class of approximation to MWM is *maximal matching* algorithms. A Maximal Match is a matching where no new edges can be added to the matching. The complexity of Maximal Matching algorithm is O(N) [13] but this class of algorithms are not throughput optimal in general.

Tassiulas has proposed a class of *Randomized Algorithms* to approximate MWM [14]. The algorithm is explained in Section II as *Algorithm Rnd3*. Our simulation results illustrate that in case of non-uniform input traffic, Tassiulas algorithm cannot follow MWM performance.

Our focus in this paper is to propose two randomized algorithms that exploit Tassiulas idea but improve its performance specifically when input traffic is non-uniform. We use MWM as the benchmark to compare the behaviour of a number of randomized algorithms as well as our proposed solution.

2. RANDOMIZED SCHEDULING ALGORITHMS

The application of randomization is widely used in new router and switch architectures [15] as well as in devising search algorithms[16]. The core idea of the randomized algorithms in our work is to choose the best (largest weight) matching among a set of possible matching in which some are randomly generated. In this section, we review the behaviour of well-known scheduling randomized algorithms at the presence of non-uniform traffic.

2.1. Definitions

a) Arrival Traffic: The arrival traffic is assumed to be i.i.d. Bernoulli and is normalized by link capacity. The parameter $\sigma \in (0,1)$ refers to the normalized load.

b) Non-Uniform Traffic: In this paper we aim to analyse the characteristics of randomized algorithms at the presence of non-uniform traffic. To model non-uniform traffic, similar to [6], we assume that input port *i* has packets only for output ports *i* and *(i+1)* mod *N*.

c) The VOQ switch is modelled by a bipartite weighted graph *G* with *2N* vertices and N^2 edges (See Figure 2). A Matching in *G* is a feasible schedule for the switch. Therefore, the terms matching and schedule is used interchangeably in this paper.

d) Associated to the j^{th} queue in input i^{th}, $1 \leq j, i \leq N,$ we define Q_{ij} equal to the length of corresponding queue. A Matrix Q_{N*N} with elements Q_{ij} is considered as the queue length matrix. Q_{ij} is the weight of edge *(i,j)* in matrix *G*.

e) A matching matrix *M* is defined such that m_{ij} is set to 1 if and only if input *i* to output *j* edge, that is edge *(i,j)* is available in the matching. The other elements of *M* is set to zero. It is notable that the total number of possible matchings in *G* is *N!*.

f) The weight of a Matching *M* is defined as the sum the weights of its edges,

$$M.*Q = \sum_i \sum_j m_{i,j} Q_{i,j} \tag{1}$$

The schedule under MWM algorithm is defined by

$$S_{MWM} = \underset{M \in \varphi}{argmax}\, M.*Q \tag{2}$$

where φ is the set of all possible schedules.

2.2. Algorithm Rnd1

In this algorithm, the required matching at each time slot is selected randomly from the set of all possible schedules by using uniform distribution.

```
$Algorithm: Rnd1$
$S_{Rnd1} = Random(\varphi)$
```

where the function *Random(φ)* begets the uniformly random selected schedule from the set φ.

2.3. Algorithm Rnd2

In *Rnd2* more than one random schedule is selected and the one with the largest weight is chosen. In other words, in *Rnd2*, the algorithm *Rnd1* is run for D times (D>1) and selects the best extracted solution. D is a parameter of the algorithm.

```
$Algorithm: Rnd2$
$For\ i = 1: D$
    $T(i) = S_{Rnd1}$
$S_{Rnd2} = \underset{T(i)}{\mathrm{argmax}}\, T(i).*Q$
```

2.4. Algorithm Rnd3

This algorithm is proposed by Tassiulas [14] and has been recognised as a stable strategy that achieves the entire throughput region.

In *Rnd3*, a random schedule is generated at time slot *t* and its weight is compared with the previous time slot schedule. The largest weight schedule is designated as the current time slot schedule. This is the first algorithm that uses memory (previous state) in its procedure. The performance of the stated randomized algorithms is studied in section IV.

$Algorithm: Rn$

3. PROPOSED ALGORITHMS

As stated before, the substantial challenge of MWM algorithm is its complexity. The proposed solutions in literature lead to less complex algorithms in cost of less throughput and more delay when traffic is not uniformly distributed among output ports. We consider the characteristics of non-uniform traffic to design two algorithms which are named Rnd4 and Rnd4rm. The details of the proposed algorithms are as follows.

3.1. Algorithm Rnd4

The foundation of *Rnd4* is similar to *Rnd3*. The only difference refers to the selection of random schedule at current time slot. In *Rnd4*, construction of the random schedule follows a smart procedure to take the non-uniform features of the traffic in to consideration. The main idea is to keep active the heavy edges from previous time slot in construction of the new matching. To this end, a parameter $0 < \eta < 1$ is defined as the input value to the algorithm. The edges in previous time slot matching are arranged in descending order of their weights. A collection of heavy edges are selected from the top of the ordered set such that the sum of their weights are larger than η times the schedule weight. The remaining edges are chosen randomly. Thus, a random schedule is created that keeps the heavy edges of the previous schedule.

```
$Algorithm: Rnd4$
Arrang the edges of $S_{Rnd4(t-1)}$in descending order of
        their weights, name the ordered set $\varphi_O$
            $Create\ Matrix\ T\ as\ follows:$
                $t(i,j) = 0, \quad \forall i, \forall j$
    $loop: \ Choose\ the\ first\ edge\ (i,j)\ from\ \varphi_O$
                $t_{i,j}(t) = m_{i,j}(t-1)$
    $if\ [\sum_i \sum_j t_{i,j}(t) * Q(t)_{i,j} > \eta M(t-1).* Q(t)]$
                    $then\ Breake.$
                $else\ \ \varphi_O \leftarrow \varphi_O - (i,j)$
                    $Goto\ loop$                              1
$S_{Rnd4}(t) = Fill\ (T)$
```

Where the function *Fill* keeps all the non-zero (one) elements of T and adds to it randomly generated ones to make a complete matching.

3.2. Algorithm Rnd4rm

This algorithm is the combination of *Rnd2* and *Rnd4*. It promotes *Rnd4* by keeping D previous schedules in to account as shown below.

```
$Algorithm: Rnd4rm$
$For\ i = 1: D$
    $T(i) = S_{Rnd4}$
$S_{Rnd4rm} = \underset{T(i)}{\operatorname{argmax}} T(i).* Q$
```

In this section, it is explained the results of research and at the same time is given the comprehensive discussion. Results can be presented in figures, graphs, tables and others that make the reader understand easily [2], [5]. The discussion can be made in several sub-chapters.

4. SIMULATION RESULTS AND DISCUSSION

To demonstrate the simulation results, MATLAB software has been used in this article. In this section, the sum of queue lengths of VOQ switch versus increasing load is depicted as the output of different simulation scenarios. The value of each point is the average of 1000 iteration of the simulation. Each iteration consists of three parts, a) packet entrance, b) packet schedule and c) packet departure. The packet size is fixes and equals to one time slot (= one iteration). The number of switch ports N=8. The parameter D in *Rnd2* and *Rnd4rm* is set to 8 and $\eta = 0.5$. The generated traffic is non-uniform with different values of σ according to definitions 1 and 2 in Section II. The performance of *Rnd1*, *Rnd2*, *Rnd3*, *Rnd4*, *Rnd4rm*, *MWM*, *Laura* and *Serena* scheduling algorithms are compared in this section.

Figure 3 indicates the superior performance of *MWM* over *Rnd1* and *Rnd2*. It is also observed that as it was expected *Rnd2* outperforms *Rnd1*. The reason is that Rnd2 is the same as Rnd1 which executes for D times. The better performance of *Rnd2* is achieved in cost of increased complexity by a factor of D. It can be observed by comparison of Figure 4 to Figure 3 that the performance of Rnd4 and Rnd3 are much improved with respect to Rnd2. The magic for this is the role of utilizing memory in the algorithms. In both Rnd3 and Rnd4, other than randomly generated matching, previous time slot(s) schedules have also a significant role in current time slot schedule. Instead of choosing between previous schedule and a randomly generated schedule as it is done in Rnd3, In Rnd4 a combination of these two schedules are generated. This procedure makes Rnd4 a superior over Rnd3. By utilizing the technique of Rnd2 in Rnd4 and revise it to Rnd4rm, our proposed algorithm become closer to MWM curve as shown in Figure 4.

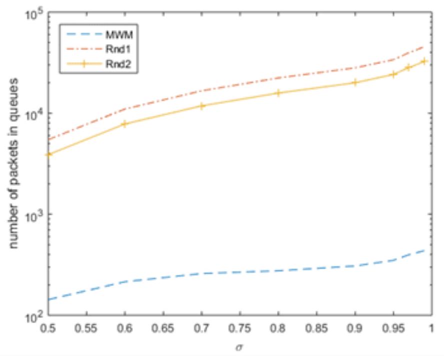

Figure 3. Comparison of Rnd1, Rnd2 and MWM algorithms

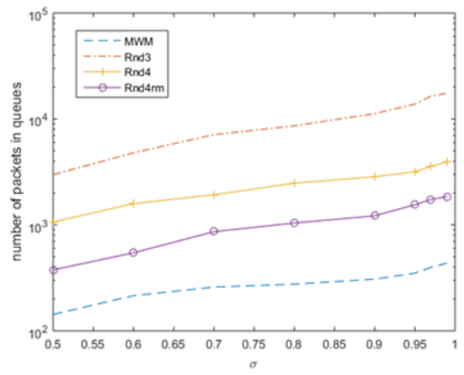

Figure 4. Comparison of Rnd3, Rnd4, Rnd4rm and MWM algorithms

In what follows, we have evaluated Rnd4rm algorithm by comparing it with two well-known randomized algorithms, Laura and Serena [6]. In Figure 5, it is noticeable that Rnd4rm outperform Laura and Serena. In Laura, random schedule at current time slot is generated iteratively to keep a fraction (η) of previous time slot matching weight. It is not necessarily guaranteed that Laura keeps the heavy weight edges in the new matching. However, in Rnd4rm, a number of largest weight edges are deterministically kept in new schedule such that a fraction (η) of the previous time slot matching weight is achieved. By this policy, Rnd4rm outperforms Laura. As the arrival traffic is non-uniform, it can be observed in Figure 2 that Serena, in which the random arrival of packets are utilized as the source of randomness, cannot compete Rnd4rm. Our proposed algorithms contain a parameter η. To see the impact of this parameter, we have run the simulation for different values of η∈{0.1,0.3,0.5,0.7,0.9}. The simulation results are plotted in the Figure 6. It shows that for η=0.1 and η=0.9 the average number of packets in queues are high compared to moderate values of η. It is observed that η=0.5 is the best choice for this parameter. The reason for this behavior is that small values of η in Rnd4rm makes it ineffective within the operation of the algorithm. In other words, a few number of heavy edges would be selected from previous schedule when η is small. On the other hand, as η approaches 1 (η→1), the advantage of random selection is vanished such that for η=1 the current schedule would be the same as the previous schedule. This trade off leads us to choose η=0.5 in our simulation setup.

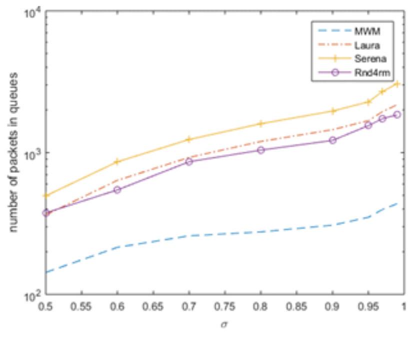

Figure 5. Comparison of Laura, Serena, Rnd4rm and MWM algorithms

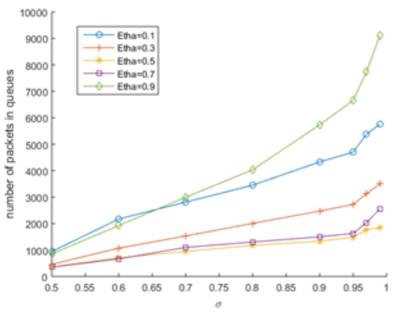

Figure 6. The impact of parameter η

5. CONCLUSION

We have considered the problem of scheduling in VOQ switch architecture. Focusing on randomized scheduling algorithms, the performance of different algorithms in this class has been evaluated for non-uniform input traffic. The bench mark for our comparison is the well-known throughput optimal MWM algorithm. We have compared the performance of a number of scheduling algorithms to MWM. The poor performance of the previously proposed solutions at the presence of non-uniform traffic was observed in simulation results. It motivates us to propose two smart scheduling algorithms with the aim of utilizing randomization and combine its output to a memory of previous schedules. Simulation results reveal that our proposed algorithms outperform Laura and Serena, two other scheduling algorithms, in terms of packet delay.

REFERENCES

[1] K. Yoshigoe and K. J. Christensen, "An evolution to crossbar switches with virtual output queuing and buffered cross points," *IEEE Netw*., vol. 17, no. 5, pp. 48–56, Sep. 2003.

[2] M. Karol, M. Hluchyj, and S. Morgan, "Input Versus Output Queueing on a Space-Division Packet Switch," *IEEE Trans. Commun*., vol. 35, no. 12, pp. 1347–1356, Dec. 1987.

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

[4] Keslassy and N. Mckeown, "*Analysis of scheduling algorithms that provide 100 % throughput in input-queued switches,*" vol. 9030.

[5] M. Andrews, K. Kumaran, K. Ramanan, A. Stolyar, R. Vijayakumar, and P. Whiting, "Scheduling in a queuing system with asynchronously varying service rates," *Probab. Eng. Informational Sci.*, vol. 18, no. 2, pp. 191-217, 2004

[6] P. Giaccone, B. Prabhakar, and D. Shah, "Randomized scheduling algorithms for high-aggregate bandwidth switches," *IEEE J. Sel. Areas Commun*., vol. 21, no. 4, pp. 546–559, 2003.

[7] N. Mckeown and S. Member, "*The iSLIP Scheduling Algorithm for Input-Queued Switches*," vol. 7, no. 2, pp. 188–201, 1999.

[8] M. A. Marsan, A. Bianco, E. Leonardi, and L. Milia, "RPA: a flexible scheduling algorithm for input buffered switches," *IEEE Trans. Commun*., vol. 47, no. 12, pp. 1921–1933, 1999.

[9] D. Shah and M. Kopikare, "*Delay bounds for approximate maximum weight matching algorithms for input queued switches*," Proceedings.Twenty-First Annu. Jt. Conf. IEEE Comput. Commun. Soc., vol. 2, pp. 1024–1031, 2002.

[10] E. Leonardi, M. Mellia, F. Neri, M. A. Marsan, D. Elettronica, and P. Torino, "*Bounds on Average Delays and Queue Size Averages and Variances in Input-Queued Cell-Based Switches,"* pp. 1–9, 2001.

[11] M. J. Neely, M. J. Neely, E. Modiano, and C. E. Rohrs, "*Tradeoffs in delay guarantees and computation complexity for $n\times n$ packet switches*," in Proc. of Conf. on Information Sciences and Systems, Princeton, 2002

[12] M. J. Neely, E. Modiano, and Y.-S. Cheng, "Logarithmic delay for N*N packet switches under the crossbar constraint,*" IEEE/ACM Trans. Netw.*, vol. 15, no. 3, pp. 657–668, 2007.

[13] T. Weller and B. Hajek, "Scheduling nonuniform traffic in a packet-switching system with small propagation delay," *IEEE/ACM Trans. Netw.*, vol. 5, no. 6, pp. 813–823, 1997.

[14] L. Tassiulas, "Linear complexity algorithms for maximum throughput in radio networks and input queued switches," in Proceedings of *IEEE INFOCOM*, vol. 2, pp. 533–539, 1998.

[15] M. Gao, S. Li, and B. Xiao, "The Application Research of Randomized Network Coding in ForCES Router," *TELKOMNIKA Indones. J. Electr. Eng.*, vol. 11, no. 2, pp. 774–782, 2013.

[16] V. Behravesh and S. M. R. Farshchi, "A Novel Randomized Search Technique for Multiple Mobile Robot Paths Planning In Repetitive Dynamic Environment," *International Journal Robot and Automation*, vol. 1, no. 4, pp. 214–222, 2012.