Journal of Southeast University (English Edition) Vol. 21, No. 1, pp. 6 - 10 Mar. 2005 ISSN 1003—7985

Randomized scheduling algorithm for input-queued switches

Wu Jun Luo Junzhou

(Department of Computer Science and Engineering, Southeast University, Nanjing 210096, China)

Abstract: The sampling problem for input-queued (IQ) randomized scheduling algorithms is analyzed. We observe that if the current scheduling decision is a maximum weighted matching (MWM), the MWM for the next slot mostly falls in those matchings whose weight is closed to the current MWM. Using this heuristic, a novel randomized algorithm for IQ scheduling, named genetic algorithm-like scheduling algorithm (GALSA), is proposed. Evolutionary strategy is used for choosing sampling points in GALSA. GALSA works with only $O(N)$ samples which means that GALSA has lower complexity than the famous randomized scheduling algorithm, APSARA. Simulation results show that the delay performance of GALSA is quite competitive with respect to that of APSARA.

Key words: switches; input-queued; randomized algorithm

Traditional switching fabrics assumed an output-queued (OQ) architecture. Output-queued switches are appealing because they have optimal delay-throughput performance for all traffic distributions. However, OQ switches are not scalable for the N-times speed-up factor. WWW and other multimedia applications have dramatically increased and will continue to increase the traffic over the Internet, which demands high performance and scalable switches. As a result, input-queued (IQ) switch architectures have recently received increasing attention.

With respect to OQ architectures, IQ has the advantage that both the switching fabric and input queues can work at the line rate. Thus IQ is more scalable than OQ. However, IQ has a critical drawback: the throughput is limited to 58.6% [1] due to the notorious head-of-line (HoL) blocking phenomena.

To overcome the drawback, virtual output queued (VOQ) technology is adopted in the IQ switch designs. In VOQ switches, each input maintains N queues, one for each output. By using VOQ, HOL can be eliminated because all cell-switching probabilities in one input port are presented on heads of queues. Therefore, VOQ switches with elaborate designed schedulers can achieve 100% throughput. It has been proved that by using some maximum weight matching (MWM) algorithms (such as LQF, LPF) [2, 3] 100% throughput can be achieved for any i. i. d arrivals.

Received 2004-06-16.

Foundation items: The National Basic Research Program of China (973 Program) (No. G1998030405), the Foundation of Excellent Doctoral Dissertation of Southeast University (No. YBJJ0408).

Biographies: Wu Jun (1970—), male, graduate; Luo Junzhou (corresponding author), male, doctor, professor, jluo@seu.edu.cn.

However, the MWM algorithm has a complexity of $O(N^3)$; even the MNCM algorithm[4] proposed recently also had a complexity $O(N^{2.5})$. It is too high to be practical.

Over the past few years, considerable work has been done in this area. A number of practical algorithms have been proposed in the literature. iSLIP[5] and FIRM[6] are round-robin based algorithms which use maximal matching instead of maximum matching. They have lower complexity than MWM and can be executed in parallel. But they suffer poor performance under nonuniform traffic. Even EDRR[7], a modified round-robin algorithm, is not satisfactory under nonuniform arrivals.

Randomized algorithms are more suitable for high aggregate bandwidth IQ scheduling due to their intrinsic parallelism. Tassiulas proposed a simple randomized IQ scheduling algorithm[8] (we name it TRA) and proved that the algorithm can achieve 100% throughput under any i. i. d arrivals. However, TRA needs hardware to support random matching, which is difficult to implement. Giaccone, et al. [9] proposed a derandomized TRA by using Hamiltonian walk instead of random matching. But the performance of TRA is poor in terms of delay-throughput. Giaccone, et al. proposed two other randomized algorithms, APSARA[9] and SERENA[10]. They perform much better than iSLIP for nonuniform traffic models. However, APSARA needs $O(N^2)$ samples to achieve good performance. It is too expensive for hardware implementation. SERENA has $O(N\log N)$ complexity but can hardly be executed in parallel. In this paper, we propose a novel randomized algorithm named genetic algorithm-like scheduling algorithm (GALSA). Simulation results

show that by computing only $O(N)$ samples GALSA outperforms previous major algorithms.

1 Background Knowledge

The IQ scheduling problem for an $N \times N$ IQ switch can be viewed as a bipartite graph matching problem, where each part contains N nodes, and one part corresponds to the input ports, while the second part corresponds to the output ports. The requests from input ports to corresponding output ports form the edges of the bipartite graph. A matching can be represented as a matrix $\boldsymbol{S} = [S_{ij}]$, where each column or row has exactly one element 1 and all others 0s. $S_{ij} = 1$ means that input port i is matched with output port j. Obviously, $\boldsymbol{S}$ can be represented equivalently as a permutation π via the equation $\pi(i) = j$ iff $S_{ij} = 1$. Let $Q_{ij}(t)$ denote the queue length of VOQ_{ij} at time t. The weight of matching $\boldsymbol{S}$ at time t is defined as $W(\boldsymbol{S}, t) = \sum_{i,j} S_{ij} Q_{ij}(t)$. $\boldsymbol{S}^*(t)$ is used to denote the corresponding maximum weight matching at time t.

There are $N!$ matchings of an $N \times N$ switch. The $N!$ matchings can be organized as a complete graph, with each node corresponding to a distinct matching. Let $Z(t)$ denote the status of Hamiltonian walk on this graph at time t (refer to Ref. [1] for more details). The optimal scheduling algorithm must work on the whole state space. Randomized algorithms, unlike the optimal algorithm such as MWM, compute a few samples rather than search the whole state space. TRA is a very simple example. TRA determines the matching at time $t+1$ by the following equation:

$$\boldsymbol{S}(t+1) = \arg \max_{\boldsymbol{S} \in \{\boldsymbol{S}(t), R(t+1)\}} W(\boldsymbol{S}, t+1) \quad (1)$$

where $R(t+1)$ is chosen uniformly at random from the whole state space. TRA achieves 100% throughput under any i. i. d arrival by evaluating only two samples, but it suffers from poor performance in terms of delay-throughput.

The performance of randomized algorithms is determined by sampling. To acquire good samples, the notation, neighbor of a matching, was introduced in Ref. [1]. A matching $\boldsymbol{S}'$ is called a neighbor of $\boldsymbol{S}$ iff $\boldsymbol{S}'$ has only two input-output pairs different from $\boldsymbol{S}$. The set of all neighbors of a matching $\boldsymbol{S}$ is denoted $N(\boldsymbol{S})$. Because at most one cell can arrive at or leave from one input port per slot, the VOQ length changes very little during successive time slots. It hints that $\boldsymbol{S}^*(t+1)$ likely exists in $N(\boldsymbol{S}^*(t))$. This observation leads to APSARA. The matching of APSARA at time $t+1$ is given by

$$\boldsymbol{S}(t+1) = \arg \max_{\boldsymbol{S}' \in M(t+1)} W(\boldsymbol{S}', t+1) \quad (2)$$

where $M(t+1) = N(\boldsymbol{S}(t)) \cup \boldsymbol{S}(t) \cup Z(t+1)$. The number of samples used by APSARA is $O(N^2)$. Computing the weights of all $O(N^2)$ neighbors requires a lot of space in hardware for large values of N.

To overcome this, a tradeoff between performance and complexity was made in Ref. [1]. Let $N_K(\boldsymbol{S}(t))$ denote K elements picked uniformly at random from the set $N(\boldsymbol{S}(t))$. The $\boldsymbol{S}(t+1)$ formula for APSARA-R(K) is similar to Eq. (2), with replacing $N(\boldsymbol{S}(t))$ by $N_K(\boldsymbol{S}(t))$. In this paper we especially use APSARA-R to replace APSARA-R(N), where N is the size of switches.

From TRA and APSARA, it is easy to see that: ① Randomized algorithms for IQ scheduling are simple to implement; ② Randomized algorithms are apt to be executed in parallel because the evaluation of all samples can be done concurrently. Therefore, the randomized algorithm is a promising technique for high aggregated bandwidth IQ scheduling, which is challenging because there is either too little time or too much work to do.

2 GA-Like Scheduling Algorithm

As mentioned in section 1, sampling is a key step of randomized algorithms for IQ scheduling. APSARA is somewhat successful because its sampling method benefits from "memory feature" of the high-speed switch scheduling problem. The so-called memory feature is the fact that the VOQ length changes very little during successive time slots. Let $M(\boldsymbol{S}(t), d)$ denote the set of matchings whose element $\boldsymbol{S}'(t)$ satisfies the following formula:

$$|W(\boldsymbol{S}(t), t) - W(\boldsymbol{S}'(t), t)| \leqslant d \quad (3)$$

In fact, the memory feature tells us that picking samples from $M(\boldsymbol{S}(t), d)$ for some small number d can get $\boldsymbol{S}^*(t+1)$ with bigger probability than picking the same number samples from the whole state space. Thus, the memory feature can be used for decreasing the number of samples needed by randomized algorithms for IQ scheduling.

Given a number d, elements of $N(\boldsymbol{S}(t))$ belong to $M(\boldsymbol{S}(t), d)$, but perhaps there are some elements of $M(\boldsymbol{S}(t), d)$ which do not belong to $N(\boldsymbol{S}(t))$. For example, let the VOQ state of a 4×4 switch at time $t+1$ and the matching at time t be as follows:

$$\mathrm{VOQ} = \begin{bmatrix} 3 & 4 & 0 & 0 \\ 0 & 3 & 4 & 0 \\ 0 & 0 & 3 & 4 \\ 3 & 0 & 0 & 3 \end{bmatrix}, \quad \boldsymbol{S}(t) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Considering the MWM matching at time $t+1$, $\boldsymbol{S}^* = \{\pi(1) = 2, \pi(2) = 3, \pi(3) = 4, \pi(4) = 1\}$. It belongs to $M(\boldsymbol{S}(t), 3)$ but not in $N(\boldsymbol{S}(t))$. This implies that the sampling method used by APSARA may miss the

MWM. Just for this reason, APSARA needs many samples to achieve good performance.

However, direct computing $M(\boldsymbol{S}(t), d)$ from (3) is very difficult. Meanwhile, the number d is also difficult to determine. If d is too small, MWM will often be missed, and if d is too large there will be too many samples. To get samples close to $M(\boldsymbol{S}(t), d)$, we propose an adaptive algorithm by using the evolutionary strategy. Before presenting the algorithm we introduce some notations used by the algorithm. Let $M_K(t)$ denote the set of K matchings at time t, which are initially chosen arbitrarily from the whole state space. Let $N_1(M_K(t))$ denote the matching set that is given by the following formula:

$$N_1(M_K(t)) = \bigcup_{\boldsymbol{S} \in M_K(t)} N_1(\boldsymbol{S}) \tag{4}$$

where $N_1(\boldsymbol{S})$ denotes a matching picked uniformly at random from all neighbors of $\boldsymbol{S}$. Let $M(t)$ denote $M_K(t) \cup N_1(M_K(t))$, obviously $N_1(M_K(t))$ has K elements and the cardinality of $M(t)$ is $2K$.

Our algorithm is as follows:

① Determine $N_1(M_K(t+1))$ by Eq. (4), and then determine $M(t+1)$.

② For every $\boldsymbol{S} \in M(t+1)$, compute the weight $W(\boldsymbol{S}, t+1)$.

③ $\boldsymbol{S}(t) = \arg \max_{\boldsymbol{S}' \in M(t+1)} W(\boldsymbol{S}', t)$.

④ Evolve the $M_K(t+1)$ to $M_K(t+2)$. Namely, for every $\boldsymbol{S} \in M_K(t)$ and its counterpart $N_1(\boldsymbol{S})$ which belongs to $N_1(M_K(t))$, if $W(\boldsymbol{S}, t) < W(N_1(\boldsymbol{S}), t)$, replace $\boldsymbol{S}$ with $N_1(\boldsymbol{S})$ in $M_K(t+2)$.

The output of the algorithm at the time slot t is $\boldsymbol{S}(t)$ which is used for configuring the cross-bar switch. Step ④ of the algorithm is the evolutionary strategy. Elements in $M_K(t)$ may be very different at time 0, but after some time t, or say, by t times evolution, the weight of element in $M_K(t)$ is close to $W(\boldsymbol{S}(t), t)$. For similar consideration of APSARA, the weight of elements in $N_1(M_K(t+1))$ is also close to $W(\boldsymbol{S}(t), t)$. Thus, we can get $2K$ samples which belong to $M(\boldsymbol{S}(t), d)$ for some number d though we do not know d. In terms of the memory feature, the heaviest matching in $M_K(\boldsymbol{S}(t))$ is very likely an $\boldsymbol{S}^*(t+1)$. It means that our algorithm has good delay-throughput performance, which is verified by the simulation results in section 4. Because the evolutionary strategy is similar to the genetic algorithm, we call the algorithm as the genetic algorithm-like scheduling algorithm (GALSA). In the rest of this paper, we will use GALSA(L) to denote the GALSA with L elements in $M(t)$ or $L = 2K$. K is an adjustable parameter of the algorithm, which is often determined by hardware space constraints. Generally speaking, making L equal to the number of input ports of a switch is suggested for moderate size switches.

3 Complexity and Implementation

The main computation of GALSA is step ③. If the weight computation of $2K$ elements in $M_K(t)$ executes concurrently, GALSA needs $O(1)$ weight computing and $O(1)$ arbitration, which is the same as APSARA-R($2K$). The arbitration is a procedure that creates a tournament tree of $2K$ numbers corresponding to $2K$ weights of elements in $M_K(t)$ as shown in Fig. 1. Obviously, step ④ does not increase any complexity of the algorithm, because the information used by step ④ is a by-product of the arbitration. We only need to make a little modification to the arbitrator.

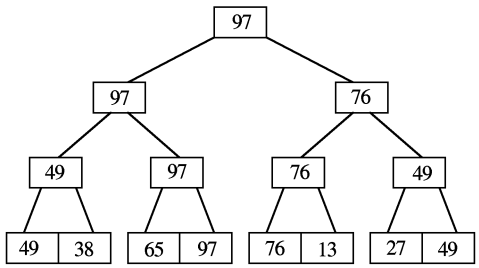

Fig. 1 An example of a tournament tree with eight leaf nodes (The root of the tree is the output of the arbitration. And the output of the first comparison can be used for step ④ of GALSA)

Fig. 2 gives a schematic scheduler for GALSA. The scheduler contains an arbiter and K units. For every matching in $M_K(t)$, there is a corresponding unit. Every unit has two weight calculators, one for the matching $\boldsymbol{S}$ in $M_K(t)$ and the other for $N_1(\boldsymbol{S})$. K units work concurrently requiring $2K$ weight calculators which also work in parallel.

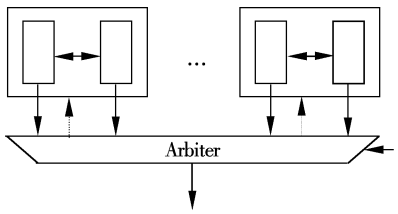

Fig. 2 A schematic scheduler for implementation of GALSA

4 Simulation Results

Before presenting the performance of GALSA, we outline the simulation setting. The simulation setting here is similar to the one used in Ref. [11].

The switch used in our simulation is a 32×32 switch. We assume that all input/output line rates are equal, and that only unicast traffic flows are present. In the IQ switch, each input queue VOQ_{ij} has finite length 2×10^5. When a cell directed to output j arrives

at input i, if queue VOQ_{ij} is full, the cell is dropped. No buffer sharing among queues is allowed.

Traffic Scenarios All inputs are equally loaded on a normalized scale, and $\rho \in (0,1)$ denotes the normalized load. The arrival process is Bernoulli i. i. d.

1) Uniform scenario In this case $\lambda_{ij} = \rho/N \forall i,j$, where λ_{ij} denotes the cell rate which arrives at input port i and directs to output j.

2) Diagonal scenario In this case $\lambda_{ii} = 2\rho/3$ $\forall i$, $\lambda_{ii+1} = \rho/3$, and $\lambda_{ij} = 0$ for all other i and j. This is a very skewed loading, in the sense that input i has cells only for outputs i and $i+1$. We use this traffic to evaluate the performance of GALSA under nonuniform arrivals.

Performance Measures We compare the queue length induced by different algorithms, the delay can be computed using Little's Law. Here the queue length is the sum of all VOQ lengths in one input port. For every algorithm we run 10^6 time slots per load.

Fig. 3 compares the average queue lengths induced by APSARA, GALSA and SERENA under diagonal traffic. As can be seen, GALSA(32) outperforms APSARA-R in all loads. And GALSA(32) performs better than APSARA in low loads and high loads. SERENA is competitive with APSARA for low loads. However, the predominance of SERENA does not exist when it compares with GALSA.

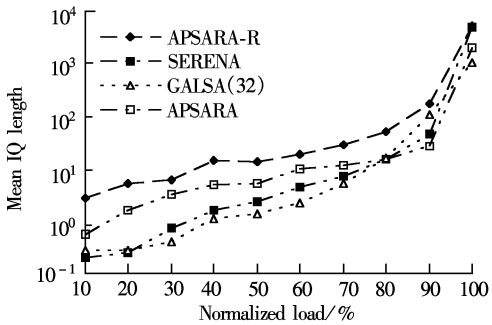

Fig. 3 Mean IQ length for randomized algorithms under diagonal traffic

The cell delay of SERENA is a little smaller than the one of GALSA only in the loads between (0.8, 0.9). In Fig. 4 we compare the performance of APSARA with GALSA more extensively. GALSA with 64 samples performs better than APSARA in almost all loads, where APSARA needs $\binom{N}{2} = 496$ samples. GALSA(L) outperforms APSARA(L) with other value of L. This shows that evolutionary strategy is useful for designing randomized algorithms for IQ scheduling problem.

Fig. 5 compares the performance of iSLIP, EDRR, FIRM and GALSA under diagonal traffic. Although EDRR improves a little in terms of delay-throughput

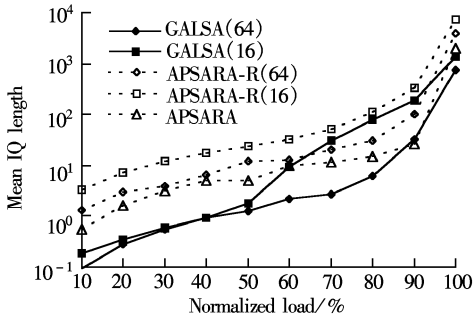

Fig. 4 Mean IQ length for APSARA-R(L) and GALSA(L) with different parameters L under diagonal traffic

on iSLIP and FIRM, it also suffers from big delay in high load when compared with GALSA, SERENA and APSARA. This maybe implies that approximate MWM algorithms are better than approximate maximal algorithms for nonuniform arrivals.

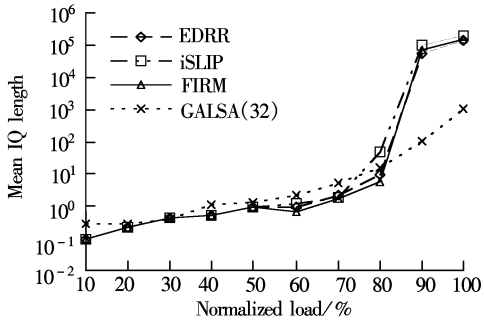

Fig. 5 Mean IQ length for RR algorithms and GALSA under diagonal traffic

Finally, we compare the performance of the algorithms referred in this paper under uniform traffic (see Fig. 6). All these algorithms are good in terms of delay. The best one is SERENA, because it merges the arrival pattern and the matching of last slot, which takes merits from both randomized algorithms and deterministic algorithms. However, it is difficult to implement in parallel, which limits its applications in very high bandwidth switches. GALSA performs based algorithms in high load but round-robin based better than round-robin algorithms are a little better than our algorithm in lower load. Because APSARA samples in neighbors of last matching are limited in

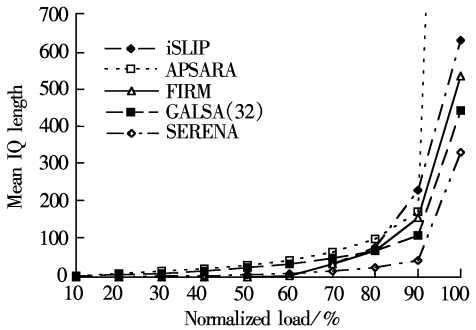

Fig. 6 Mean IQ length for uniform traffic

very local area, APSARA performs the worst in all these algorithms under uniform i. i. d traffic.

5 Conclusion

This paper has proposed a new randomized algorithm for the high bandwidth IQ scheduling problem. The main idea of the algorithm is to use evolutionary strategy and the memory feature. Simulation results show that the sampling method of the new algorithm overcomes the drawback of APSARA, which makes GALSA outperform some major previous algorithms presented in the literature.

References

[1] Karol M J, Hluchyj M, Morgan S. Input versus output queueing on a space-division packet switch [J]. *IEEE Transactions on Communications*, 1987, **35**(12): 1347 – 1356.

[2] Mckeown N, Mekkittikul A, Anantharam V, et al. Achieving 100% throughput in an input-queued switch [J]. *IEEE Transactions on Communications*, 1999, **47**(8): 1260 – 1267.

[3] Mekkittikul A, Mckeown N. A practical scheduling algorithm to achieve 100% throughput in input-queued switches[A]. In: Guerin R, ed. *Proceedings of the IEEE INFOCOM* [C]. San Francisco: IEEE Computer Society Press, 1998. 792 – 799.

[4] Tabatabaee V, Tassiulas L. MNCM a new class of efficient scheduling algorithms for input buffered switches with no speedup[A]. In: Matta I, ed. *Proceedings of the IEEE INFOCOM*[C]. San Francisco: IEEE Communications Society, 2003. 1406 – 1413.

[5] McKeown N. The iSLIP scheduling algorithm for input-queued switches [J]. *IEEE Transactions on Networking*, 1999, **7**(2): 188 – 201.

[6] Serpanos D N, Antoniadis P I. FIRM: a class of distributed scheduling algorithms for high-speed ATM switches with multiple input queues[A]. In: Katzela I, ed. *Proceedings of the IEEE INFOCOM*[C]. Tel Aviv: IEEE Communications Society, 2000. 548 – 555.

[7] Li Y, Panwar S, Chao H J. The dual round robin matching switch with exhaustive service[A]. In: Gunner C, ed. *Proceedings of IEEE Workshop on High Performance Switching and Routing*[C]. Kobe, Japan: IEEE Communications Society, 2002. 58 – 63.

[8] Tassiulas L. Linear complexity algorithms for maximum throughput in radio networks and input queued switches [A]. In: Guerin R, ed. *Proceedings of the IEEE INFOCOM* [C]. San Francisco: IEEE Computer Society Press, 1998. 533 – 539.

[9] Giaccone P, Shah D, Prabhakar B. An implementable parallel scheduler for input-queued switches[J]. *IEEE Micro*, 2002, **22**(1): 19 – 25.

[10] Shah D, Giaccone P, Prabhakar B. An efficient randomized algorithm for input-queued switch scheduling [J]. *IEEE Micro*, 2002, **22**(1): 10 – 18.

[11] Marsan M A, Bianco A, Giaccone P, et al. Packet scheduling in input-queued cell-based switches[A]. In: Ammar M, ed. *Proceedings of the IEEE INFOCOM*[C]. Anchorage: IEEE Communications Society, 2001. 1085 – 1094.

输入队列交换机的一种随机调度算法

吴 俊 罗军舟

(东南大学计算机科学与工程系, 南京 210096)

摘要: 对输入队列随机调度算法的取样问题进行了分析, 指出由于输入队列的记忆特性, 当前时隙的调度决策若具有最大权值, 那么选取与这个最大权值相近的匹配作为下个时隙调度决策时的样点将以较大概率找到最大权值匹配. 基于此本文设计了一种新的随机调度算法 GALSA, GALSA 利用演化策略来跟踪与每个时隙决策具有相近权值的匹配点. GALSA 算法所需样点是 $O(N)$, 因此其复杂性大大低于现有随机算法 APSARA. 且仿真结果表明 GALSA 的延迟性能与 APSARA 媲美.

关键词: 交换机; 输入队列; 随机算法

中图分类号: TP301