RZ 3856 (# ZUR1311-059) 12/10/2013
Computer Science 4 pages

Research Report

Performance of Random Arbitration in Switching Fabrics

Lydia Y. Chen*, Nikolaos Chrysos*

*IBM Research – Zurich
8803 Rüschlikon
Switzerland

Performance of Random Arbitration in Switching Fabrics

Lydia Y. Chen
IBM Research Zurich Lab
4 Saeumerstrasse
Rueschlikon, Switzerland
yic@zurich.ibm.com

Nikolaos Chrysos
IBM Research Zurich Lab
4 Saeumerstrasse
Rueschlikon, Switzerland
cry@zurich.ibm.com

ABSTRACT

This paper presents a throughput analysis for switches or switching fabrics with small output buffers, where a scheduler needs to compute approximate matchings, i.e., match multiple inputs to the same output. We adopt a random arbitration scheme, which is an extension of Parallel Iterative Matching (PIM) [1, 3]. We propose a performance analysis based on a Markov Chain modeling of output credits and further devise an iterative solving procedure for it. The analytical results match very well with simulations of various switch and buffer sizes. Our analysis and experiments also show that with very modest output buffers, switches of any size can provide full throughput under uniformly distributed traffic.

1. INTRODUCTION

Modern switches and switching fabrics conventionally employ virtual output queues (VOQs) at network adapters, in order to mitigate head-of-line blocking. The core of the switch, often a crossbar, can either be bufferless or buffered. Previous research on random arbitration for bufferless crossbars found that the throughput of large switch sizes is bounded by 63 %. With the rapid advances in IC technology, crossbars containing crosspoint buffers have become popular [4] and are shown to reach 100 % throughput under uniform traffic [1]. However, their scalability is restricted by the quadratic growth of the number of crosspoint buffers, and the dimension of crosspoint buffers, which depend on the round-trip time (RTT) of VOQ-crossbars.

In this paper, we study an alternative architecture, which can reduce the buffer requirements in buffered crossbars, by having a linear growth of the number of buffers, and making the buffer size independent of the RTT. As the scheduler in such an architecture needs to match multiple inputs to the same output, we refer to such matchings as *approximate*, as opposed to the *exact* matchings required in bufferless crossbars. To our best knowledge, this architecture was mainly studied through simulations. We first extend the Parallel Iterative Matching (PIM) scheduling algorithm [1], to compute approximate matchings. Furthermore, we derive the throughput analysis using a Discrete Time Markov Chain (DTCM) modeling, which is solved by a proposed interactive procedure.

Although in this paper we consider a single-stage switch, our analysis serves as a first approximation for scheduled multi-stage switching fabrics [2].

1.1 Architecture and Random Arbitration

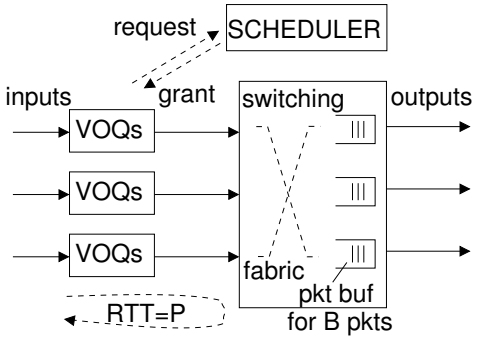


Figure 1: Switch model for N=3.

Fig. 1 depicts a $N \times N$ switch architecture, consisting of a scheduler, N output buffers with sizes of B packets, and N VOQs with sufficiently large space. We assume synchronous, slotted time, and fixed-size packets, whose transmission take one time slot. Packet arrivals are i.i.d uniformly distributed. Arriving packets issue a request to the scheduler and wait in VOQs. Upon receiving scheduler grant, packets are injected into the switching fabric and wait in the output buffers, which drain one packet per unit time. Note that when $B = 1$ packet, this model corresponds to a bufferless crossbar with VOQs.

Inputs required by scheduler are the pending VOQ requests, $pr[][]$, and the values of credit counters $cr[]$. The scheduler manages the pending requests using per-VOQ counters, and implements a credit-based flow control by using credit counters, $cr[i]$. Contention is resolved by per-output and per-input arbiters. Similar to PIM, each input and output operates in parallel. But whereas PIM computes an exact matching, our algorithm computes an *approximate match* after the following handshaking.

1. Each unmatched input j sends a request to every output i for which pending VOQ requests is greater than zero, i.e., $pr[i][j] > 0$.

2. If an output i with non-zero credit counter receives a request from m inputs, it will choose $k = \min(cr[i], m)$ of them to grant, randomly. The output will notify each input whether its request was granted.

3. If an input j receives one or more grants, it chooses

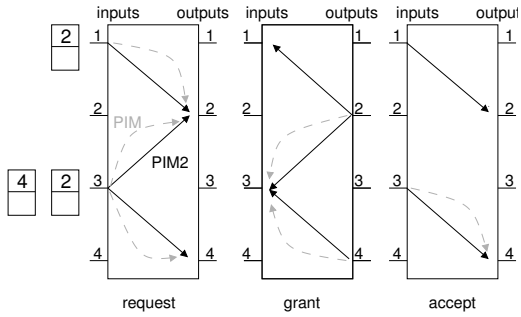


Figure 2: Generalized PIM for B=2 (PIM2); with gray dashed lines we show the selections of PIM as in [1].

one to accept randomly, and notifies that output.

At the end of this handshaking, the head-of-line packets on matched VOQ are injected and every output arbiter decreases its credit counter by the number of accept notifications it received. To render the size of buffers being independent of the RTT, we use a credit prediction scheme so that the credits allocated to inputs can be returned in advance. In particular, our scheduler increases credit counters that are below B by one, before computing a new match.

In Fig. 2, we borrow a scenario from [1] to show the steps of the generalized PIM algorithm for $B = 2$ (PIM2), assuming that all credit counters are set to 2. PIM is also depicted using gray dashed lines. In this scenario, input 1 has a packet for output 2, while input 3 has packets for outputs 2 and 4. In generalized PIM2, output 2 grants both inputs 1 and 3, whereas in PIM it has to select one, and randomly grants input 3.

In both PIM and PIM2, input 3 receives two grants, and randomly selects that from output 4. As shown in the figure, PIM2 finds a maximum-size bipartite graph matching, whereas PIM leaves input 3 unmatched.

2. THROUGHPUT ANALYSIS

We derive the maximum throughput of the aforementioned system, for a given output buffer size, B. At every time slot, an output has x credits available, $x \in [1, B]$, and randomly selects x input ports to grant. Meanwhile, an input may receive up to y grants, $y \in [0, N]$, from different outputs, only one of which can be accepted.

The maximum throughput of the system is equivalent to the average utilization of an output, as the traffic is uniformly distributed. One can further show the equivalence of the utilization of outputs and inputs. Let $U^O(T)$ and $U^I(T)$ denote the utilization of outputs and inputs measured over an interval of T consecutive time slots respectively. As output buffers (of size B) never overflow, $U^O(T)$ cannot lag behind by more than $\frac{B}{T}$. On the other hand, conservation law states that output utilization may exceed input utilization by up to $\frac{B}{T}$ (due to packets in output buffers). Therefore, we know $|U^O(T) - U^I(T)| \leq \frac{B}{T}$. As $T \to \infty$, the steady state utilization of outputs and inputs are equivalent, $U^O(T) \cong U^I(T)$. In the remainder of this paper, we denote U as the steady state utilization of the system.

When B=1, the scheduler holds exactly one credit for each output at every time slot and performs identically to PIM. Following the derivation in [3], the input utilization is equal to the probability that an input port receives at least one grant, $Pr(y \geq 1)$. This probability is equal to one minus the probability that all N outputs grant remaining $N-1$ inputs, i.e., $1 - (\frac{N-1}{N})^N$. As $N \to \infty$

$$U = Pr(y \geq 1) = 1 - \left(\frac{N-1}{N}\right)^N \cong 1 - e^{-1}. \quad (1)$$

However, when $B > 1$, it's not as straightforward as $B = 1$ for obtaining $Pr(y \geq 1)$, as the number of credits, x, available at every output, may vary between 1 and B. In steady state, each output holds an average number of credits, $\beta = E[x]$, whose derivation is detailed in subsection 2.1. As a result, we obtain

$$U = Pr(y \geq 1) = 1 - \left(\frac{\binom{N-1}{\beta}}{\binom{N}{\beta}}\right)^N = 1 - \left(\frac{N-\beta}{N}\right)^N. \quad (2)$$

One can straightforwardly obtain the upper bound of U by the following lemma:

LEMMA 2.1. *When $B > 1$,*

$$U \leq 1 - \left(\frac{N-B}{N}\right)^N.$$

PROOF. From Eq. 2, $Pr(y \geq 1)$ is monotonically increasing with β, and is maximized when $\beta = B$. As a result, $1 - (\frac{N-B}{N})^N$ is the upper bound of $P(y \geq 1)$ and U as well. □

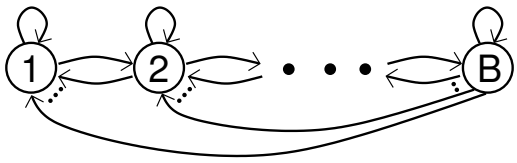


Figure 3: State transition diagram for the number of available credits at an output.

2.1 Output Credit Modeling

To derive the average number of credits at outputs, β, we model the number of credits at an output, x, as a Discrete Time Markov Chain (DTMC) process with state space, $x = \{1 \ldots B\}$.

Fig. 3 depicts the state transition diagram. Let the transition matrix be a $B \times B$ matrix, $\mathbf{P}$, where $P_{ij} = Pr\{x = j | x = i\}$ is the probability of changing from state $x = i$ to $x = j$. The state transition depends on a, denoting the number of output grants accepted by inputs. At the end of the time slot, the output decreases its credit counter by one for each accepted grant. Note that if the credit update phase results in the output having less than B credits, one credit will be returned back at the beginning of the next time slot.

Assume the output is at state i, and randomly grants i inputs. If a inputs accept the grants from this output ($a \leq i$), the next state is $x = j = \min(i - a + 1, B)$. For instance, when $a = i$, i.e., all grants are accepted, in the next time slot the output will have exactly one credit, i.e., $j = 1$. On the other hand, if $a = 0$, i.e., no grant is accepted, in the next time slot the output will have $j = \min(i+1, B)$ credits. Note that when the current state has full credits, $i = B$, the next

state can also have full credits $j = B$ under two scenarios: (1) no grant is accepted by the inputs ($a = 0$), or (2) exactly one grant is accepted ($a = 1$).

We will derive P_{ij} using q_a, the probability a grant is accepted at an input. Note probability q_a depends on the particular state of each output; to avoid modeling the entire state spaces of all outputs simultaneously, we employ a numerical procedure to estimate q_a in the next subsection. Transition from state i to j implies that $a = i - j + 1$ grants were accepted and $j - 1$ $(i - a)$ rejected; therefore, $P_{ij} = \binom{i}{i-j+1} q_a^{i-j+1}(1-q_a)^{j-1}$, as shown in the first case of Eq. 3. The second case of Eq. 3 accounts for $P_{B,B}$, which is the sum of the probabilities of no grant accepted ($\binom{i}{0}(1-q_a)^B$), and one grant accepted ($\binom{i}{1}q_a^1(1-q_a)^{B-1}$). The rest of the transitions have zero probability.

$$P_{ij} = \begin{cases} \binom{i}{i-j+1} q_a^{i-j+1}(1-q_a)^{j-1} & \text{if } i <= B, j \le i+1; \\ \binom{i}{1} q_a^1(1-q_a)^{i-1} + \binom{i}{0}(1-q_a)^i, & \text{if } i = B, j = B; \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

Once $\mathbf{P}(q_a)$ is obtained, the steady state probability of available output credits, $\pi = \{\pi_1 \dots \pi_B\}$, can be computed by the following equations:

$$\boldsymbol{\pi}\mathbf{P} = \boldsymbol{\pi}, \sum_i \pi_i = 1. \tag{4}$$

The average number of available output credits at an output is thus

$$\beta = E[x] = \sum_i i\pi_i.$$

Finally, substituting β into Eq. 2, we obtain the system throughput.

A lower bound of utilization: One may also derive a lower bound of output utilization by $1-\pi_B$. When an output is at state $x \in [1, B-1]$, it implies that at least one input has accepted the output grant, and the output is busy. Now an output is at state $x = B$ under two scenarios: (1) the output is idle, e.g. all inputs rejected its grant; or (2) exactly one input has accepted its grant, and the output is busy. Thus, the probability that an output is idle is less than π_B, and

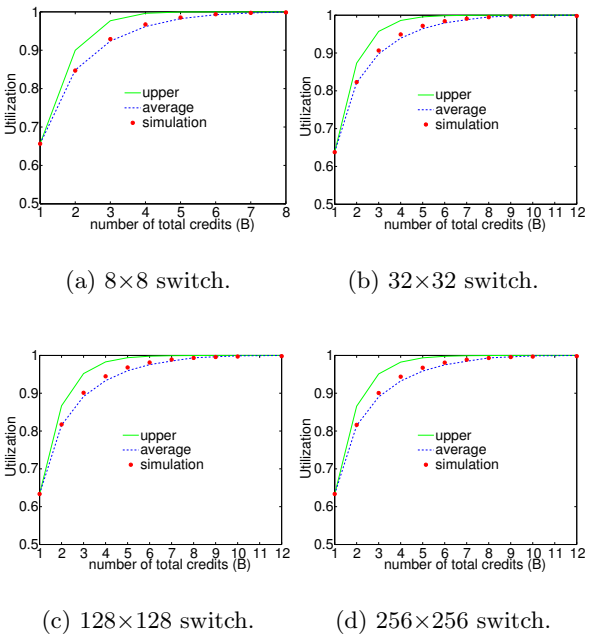
$$U \ge 1 - \pi_B. \tag{5}$$

2.2 Iterative Solving Procedure for the DTMC of Available Output Credits

To solve the DTMC of available credits at an output, one needs to obtain q_a and substitute it into Eq. 3 and Eq. 4. Instead of deriving q_a in a closed form expression, we propose using the following observation to obtain the appropriate numerical value for it. The probability of a grant being accepted by some inputs is approximately the inverse of the average number of grants that the input receives, $E[y]$. Due to the fact that outputs randomly grant inputs, in the long run $E[y] = E[x] = \beta$. Therefore, we know that

$$q_a \cong \frac{1}{\sum_i i\pi_i} = \frac{1}{\beta}. \tag{6}$$

We use an iterative solving procedure based on incremental q_a, illustrated in Algorithm 1. The algorithm initializes $q_a = 0$, iteratively obtains $\mathbf{P}(q_a)$, and solves $\beta = \sum_i i\pi$. If the relative difference between q_a and $1/\beta$, is greater than a predefined value, ϵ, the algorithm iterates with new $q_a = q_a + \delta$; otherwise the algorithm terminates. Note that values δ and ϵ may vary, and affect the quality of the solution.



(a) 8×8 switch. (b) 32×32 switch.

(c) 128×128 switch. (d) 256×256 switch.

Figure 4: Simulation v.s analytical results: throughput under different switch and buffer sizes.

Algorithm 1 Iterative Procedure: π and β

```
Initialization: q_a = 0, β = B.
while |1/β − q_a|/q_a ≤ ε do
  Let q_a = q_a + δ
  Solve π_i with P(q_a)
  Compute β = Σ_i iπ_i
end while
```

From our experience, we find that the proposed algorithm is fairly stable with reasonable convergency when q_a is first set to 0, and then iteratively increased by δ. A variation of this iterative procedure is to update $q_a = 1/\beta$ in each iteration. Compared to increasing q_a by δ in each iteration, this alternative can potentially converge faster but, based on our numerical experiments, with less stability.

3. NUMERICAL RESULTS

We present the comparisons of analytical to simulation results with respect to different buffer sizes, B, and switch sizes, N, in Fig. 4. The simulation results presented are the average of four independent simulation runs, using different random generation seeds. Each subfigure comprises separate plots for (1)**simulation** results, (2)**average:** using Eq. 2, where β is computed through the proposed DTMC, and (3) **upper:** the upper bound of throughput from lemma 2.1. One can observe that the results from the analysis match very closely with simulated values, especially for intermediate B values, namely between 2 and 8 packets. Moreover, the difference between the upper bound and the average of

throughput diminishes with increasing B and increasing N. Note that for $B = 1$, both analysis and simulation are identical to the throughput of PIM, i.e., 0.63, for large N. We also observe that the dependency between buffer size and switch size diminishes with increasing B. For example, in order to achieve 96% utilization, an 8×8 switch needs 4 credits, whereas 32×32 needs 5 credits. But when B is greater than 8, all switch sizes can achieve $\geq$ 99% utilization. Lastly, but most importantly, it is straightforward to compute the throughput for larger switch sizes, using the proposed analytical model.

4. CONCLUSIONS

We presented a throughput analysis for switching fabrics with small output buffers that employ random arbitration to compute approximate matchings. We computed the maximum sustainable throughput from the average available credits at outputs, which is obtained through a DTMC modeling. We also devised an iterative solving procedure for the DTMC of available output credits, based on the received grants at inputs. Moreover, we provided upper and lower bounds of the throughput. Both our analysis and computer simulation results showed that, even using small output buffer sizes, using random arbitration to compute approximate matchings yields almost 100% throughput.

5. REFERENCES

[1] T. E. Anderson, S. S. Owicki, J. B. Saxe, and C. P. Thacker. High-speed Switch Scheduling for Local-Area Networks. *ACM Transactions on Computer Systems*, 11(4):319–352, 1993.

[2] N. Chrysos and M. Katevenis. Scheduling in non-blocking buffered three-stage switching fabrics. In *Proc. IEEE INFOCOM*, 2006.

[3] N. McKeown. The iSLIP Scheduling Algorithm for Input-queued Switches. *IEEE/ACM Trans. Netwokr*, 7(2):188–201, 1999.

[4] R. Rojas-Cessa, E. Oki, and H. J. Chao. On the Combined Input-crosspoint Buffered Switch with Round-Robin Arbitration. *IEEE Transactions on Communications*, 53(11):1945–1951, 2005.