Concurrent Round-Robin Dispatching Scheme in a Clos-Network Switch

Eiji Oki†* Zhigang Jing‡ Roberto Rojas-Cessa‡ H. Jonathan Chao‡
†NTT Network Service Systems Laboratories
3-9-11 Midori-cho, Musashino-shi,
Tokyo 180-8585 Japan
‡Department of Electrical Engineering
Polytechnic University, 6 Metrotech Center,
Brooklyn, NY 11201 USA

Abstract— **A Clos-network switch architecture is attractive because of its scalability. Previously proposed implementable dispatching schemes from the first stage to the second stage, such as random dispatching, are not able to achieve a high throughput unless the internal bandwidth is expanded. This paper proposes a concurrent round-robin dispatching (CRRD) scheme for a Clos-network switch, to overcome the throughput limitation of the random dispatching scheme. The CRRD scheme provides high switch throughput without expanding internal bandwidth. CRRD implementation is very simple because only simple round-robin arbiters are adopted. In CRRD, the round-robin arbiters concurrently perform the matching between requesting cells and output links in each first-stage module to dispatch the cells to available second-stage modules. We show that CRRD achieves 100% throughput under uniform traffic. When the offered load reaches 1.0, the pointers of round-robin arbiters at the first-stage and second-stage modules are effectively desynchronized and contention is avoided.**

key words: Packet switch, Clos-network switch, dispatching, arbitration, throughput

I. Introduction

As Internet traffic grows explosively, high-speed switches and routers that have over 1-Tbit/s throughput are becoming necessary. Several approaches have been presented toward high-speed packet switching systems [1], [2], [9]. Most high-speed packet switching systems adopt a fixed-size cell in the switch fabric. Variable-length packets are segmented into several fixed-sized cells when they arrive, switched through the switch fabric, and reassembled into packets before they depart.

For implementation in a high-speed switching system, there are mainly two approaches. One is a single-stage switch architecture. An example of a single-stage architecture is a crossbar switch. Identical switching elements are arranged on a matrix plane [11]. [9] presented a 320-Gbit/s crossbar switch fabric using a centralized scheduler called *i*SLIP. [3] introduced a centralized contention resolution scheme for a large-capacity crossbar optical switch. Although these approaches are effective up to a certain switch size, the complexity of the switching elements is proportional to the square of the number of switch ports. This makes it difficult to expand to a large-scale switch cost-effectively.

The other approach is to use a multiple-stage switch architecture, such as a Clos-network switch [5]. The Clos-network switch architecture, which is a three-stage switch, is very attractive because of its scalability. We can categorize the Clos-network switch architecture into two types. One has buffers to store cells in the second-stage modules, and the other has no buffers in the second-stage modules.

[1] demonstrated a gigabit ATM (Asynchronous Transfer Mode) switch using buffers in the second-stage. In this architecture, every cell is randomly distributed from the first-stage to the second-stage modules to balance the traffic load in the second-stage [1]. The purpose of implementing buffers in the second-stage modules is to resolve contention among cells from different first-stage modules [12]. If the internal speed-up factor is set to be more than 1.25, the switch provides high performance [1]. However, it requires a re-sequence function at the third-stage modules or the latter modules, because the buffers in the second-stage modules cause an out-of-sequence problem. Furthermore, as the port speed increases, this re-sequence function makes it more difficult to implement.

[6] has developed an ATM switch by using non-buffered second-stage modules. This approach is promising even if the port line speed increases, because it does not cause the out-of-sequence problem. Since there is no buffer in the second-stage modules to resolve the contention, how to dispatch cells from the first stage to the second stage becomes an important issue.

A random dispatching (RD) scheme is widely used for cell dispatching from the first stage to the second stage [6], as it is adopted in the case of the buffered second-stage modules in [1]. This is because this scheme can fully distribute traffic evenly to the second-stage modules. However, RD is not able to achieve a high throughput unless the internal bandwidth is expanded, because the contention at the second stage cannot be avoided. To achieve 100% throughput by using RD, the internal expansion ratio has to be set to about 1.6 when the switch size is large [6]. This makes it difficult to implement a high-speed switch in a cost-effective manner.

One question arises: Is it possible to achieve a high throughput by using a practical dispatching scheme, without allocating any buffers in the second stage to avoid the out-of-sequence problem and without expanding the internal bandwidth?

This paper presents a solution to this question. We introduce an innovative dispatching scheme, called the concurrent round-robin dispatching (CRRD) scheme, for a Clos-network switch. The basic idea of the novel CRRD scheme is to use the desynchronization effect [7] in the Clos-network switch. The desynchronization effect has been studied using simple scheduling algorithms as *i*SLIP [7], [10] and Dual Round-Robin Matching (DRRM) [3], [4] in an input-queued crossbar switch. CRRD provides high switch throughput without increasing internal bandwidth, while the implementation is very simple because only simple round-robin arbiters are employed. We show that CRRD achieves 100% throughput under uniform traffic.

The remainder of this paper is organized as follows. Section II explains the problem of RD. Section III introduces CRRD to overcome the problem. Section IV presents the performance study of CRRD. Section V summarizes the key points.

*This work was done while he was a Visiting Scholar at Polytechnic University, Brooklyn, NY.

II. Random Dispatching (RD) Scheme

The ATLANTA switch that was developed by Lucent Technologies used the *random* nature of its dispatching algorithm [6].

This section describes the switch model that uses the RD scheme and its performance characteristics in detail, because understanding the feature of RD is helpful to understand the concurrent round-robin dispatching (CRRD) scheme described in the next section.

A. Switch model

Figure 1 shows a three-stage Clos-network switch. The terminology used in this paper is listed as follows.

IM	Input module at the first stage.
CM	Central module at the second stage.
OM	Output module at the third stage.
n	Number of input ports/output ports in each IM/OM, respectively.
k	Number of IMs/OMs.
m	Number of CMs.
i	IM number, where $0 \le i \le k-1$.
j	OM number, where $0 \le j \le k-1$.
h	Input-port (IP)/output-port (OP) number in each IM/OM, respectively, where $0 \le h \le n-1$.
r	Central-module (CM) number, where $0 \le r \le m-1$.
$IM(i)$	ith IM.
$CM(r)$	rth CM.
$OM(j)$	jth OM.
$IP(i,h)$	hth input port at $IM(i)$,
$OP(j,h)$	hth output port at $OM(j)$.
$VOQ(i,j,h)$	Virtual output queue at $IM(i)$ that stores cells destined for $OP(j,h)$.
$L_i(i,r)$	Output link at $IM(i)$ that is connected to $CM(r)$.
$L_c(r,j)$	Output link at $CM(r)$ that is connected to $OM(j)$.

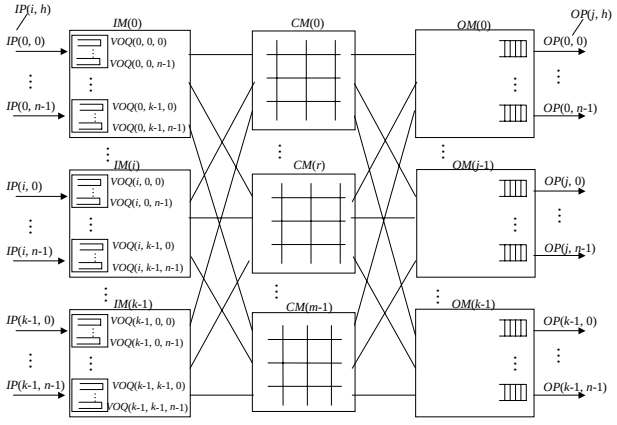

Fig. 1. Clos-network switch with virtual output queues (VOQs) in the input modules

The first stage consists of k input modules (IMs), each of which has a $n \times m$ dimension. The second stage consists of m buffer-less central modules (CMs), each of which has a $k \times k$ dimension. The third stage consists of k output modules (OMs), each of which has a $m \times n$ dimension.

An $IM(i)$ has nk Virtual Output Queues (VOQs) to eliminate Head-Of-Line (HOL) blocking. A VOQ can receive at most n cells from n input ports and send one cell to CM in one cell time slot.

A $CM(r)$ has k output links, each of which is denoted as $L_c(r,j)$. They are connected to k OMs.

An $OM(j)$ has n output ports, each of which is denoted as $OP(j,h)$ and each has an output buffer.[1] Each output buffer receives at most m cells at one time slot, and each output port at OM forwards one cell in a first-in-first-out (FIFO) manner to the output line.

B. Random Dispatching (RD) Algorithm

For dispatching from the first stage to the second stage, two phases are considered. The details of RD are described in [6]. We show an example of RD in Section II-C. In phase 1, at most m VOQs are selected as candidates and the selected VOQ is assigned to an IM output link, $L_i(i,r)$. A request that is associated with this $L_i(i,r)$ is sent from IM to CM. This matching between VOQs and output links is performed only within IM. The number of VOQs that are chosen in this matching is always $\min(d_{nev}, m)$, where d_{nev} is the number of non-empty VOQs. In phase 2, each selected VOQ that is associated with each IM output link sends a request from IM to CM. CMs respond with the arbitration results to IMs so that the matching between IMs and CMs can be done.

- Phase 1: Matching within IM
  - Step 1: At each time slot, non-empty VOQs send requests for candidate selection.
  - Step 2: $IM(i)$ selects at most m requests out of nk non-empty VOQs. For example, a round-robin arbitration can be employed for this selection [6]. Then, $IM(i)$ proposes at most m candidate cells *randomly* to $CM(r)$ by using $L_i(i,r)$.
- Phase 2: Matching between IM and CM
  - Step 1: A request that is associated with $L_i(i,r)$ is sent out to the corresponding $CM(r)$. An arbiter that is associated with $L_c(r,j)$ selects one request among k requests. A random-selection scheme is used for this arbitration. $CM(r)$ sends up to k grants, each of which is associated with one $L_c(r,j)$, to the corresponding IMs.
  - Step 2: If a VOQ at IM receives the grant from CM, it sends the corresponding cell at next time slot. Otherwise, the VOQ will be a candidate again at step 2 in phase 1 at the next time slot.

C. Performance of RD Scheme

Although RD can dispatch cells evenly to CMs, a high switch throughput cannot be achieved due to the contention at CM, unless the internal bandwidth is expanded.

Figure 2 shows an example of the throughput limitation in the case of $n = m = k = 2$. Let us assume that every VOQ is always occupied with cells. Each VOQ sends a request for a candidate at every time slot.

We estimate how much utilization an output link $OP(j,h)$ can achieve for the cell transmission. Since the utilization of every $OP(i,j)$ is the same, we focus only on

[1] We assume that the output buffer size at $OP(j,h)$ is large enough to avoid cell loss so that we can focus the discussion on the properties of dispatching schemes.

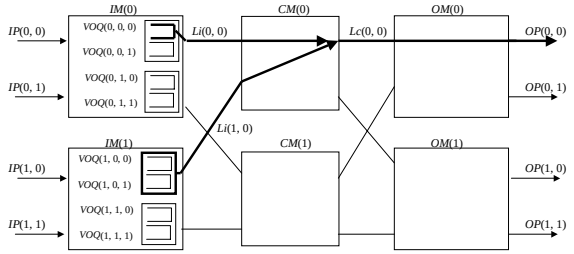

Fig. 2. Example of random dispatching (RD) scheme ($n = m = k = 2$)

a single one, $OP(0,0)$. The link utilization of $OP(0,0)$ is obtained from the sum of the cell-transmission rates of $VOQ(0,0,0)$ and $VOQ(1,0,0)$.

First, we estimate how much traffic $VOQ(0,0,0)$ can send through $CM(0)$. The probability that $VOQ(0,0,0)$ uses $L_i(0,0)$ to request for $L_c(0,0)$ is 1/4, because there are four VOQs in IM(0). Consider that $IM(1)$ sends a request for $L_c(0,0)$ using $L_i(1,0)$. If either $VOQ(1,0,0)$ or $VOQ(1,0,1)$ send a request for $L_c(0,0)$ through $L_i(1,0)$, a contention occurs with the request by $VOQ(0,0,0)$ at $L_c(0,0)$. The aggregate probability that either $VOQ(1,0,0)$ or $VOQ(1,0,1)$, among four VOQs in $IM(1)$, requests for $L_c(0,0)$ through $L_i(1,0)$ is 1/2. In this case, the winning probability of $VOQ(0,0,0)$ is 1/2. If there is no contention for $L_c(0,0)$ caused by requests in $IM(1)$, $VOQ(0,0,0)$ can always send a cell with a probability of 1.0 without contention. Therefore, the traffic that $VOQ(0,0,0)$ can send through $CM(0)$ is given as follows.

$$\frac{1}{4} \times \frac{1}{2} \times \frac{1}{2} + \frac{1}{4} \times \left(1 - \frac{1}{2}\right) \times 1.0 = \frac{3}{16} \tag{1}$$

Since $VOQ(0,0,0)$ can use either $CM(0)$ or $CM(1)$ (i.e., two CMs) and $VOQ(1,0,0)$ is in the same situation as $VOQ(0,0,0)$ (i.e., two VOQs), the total link utilization of $OP(0,0)$ is:

$$\frac{3}{16} \times 2 \times 2 = 0.75. \tag{2}$$

Thus, in this example of $n = k = m = 2$, the switch can achieve a throughput of only 75%, unless the internal bandwidth is expanded.

Next, this observation can be extended to the general case. We derive the formula that gives the maximum throughput T_{max} by using RD under uniform traffic in the following equation.

$$\begin{aligned} T_{max} = \quad & \min\left\{ \frac{m}{n} \sum_{i=0}^{k-1} \begin{pmatrix} k-1 \\ k-i-1 \end{pmatrix} \right. \\ & \left. \times \left(\tfrac{1}{k}\right)^{k-i-1} \left(1 - \tfrac{1}{k}\right)^{i} \tfrac{1}{k-i}, 1.0 \right\} \end{aligned} \tag{3}$$

Note that the factor $\frac{m}{n}$ in Eq. (3) expresses the expansion ratio.

When the expansion ratio is 1.0, (i.e., $m = n$), the maximum throughput is only a function of k. As described in the above example of $k = 2$, the maximum throughput is 0.75. As k increases, the maximum throughput decreases. When $k \to \infty$, the maximum throughput tends to $T_{max} \to 1 - \frac{1}{e} \approx 63\%$. In other words, to achieve 100% throughput by using RD, the expansion ratio $\frac{m}{n}$ has to be set to at least $(1 - \frac{1}{e})^{-1} \approx 1.582$. This makes the cost-effective implementation of the large-scale switches difficult to achieve.

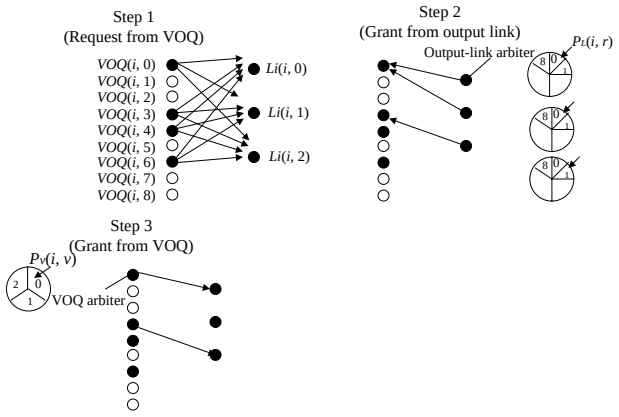

Fig. 3. Concurrent round-robin dispatching (CRRD) scheme

III. Concurrent Round-Robin Dispatching (CRRD) Scheme

A. CRRD algorithm

Figure 3 illustrates the detailed CRRD algorithm. To determine the matching between a request from VOQ(i,j,h) and the output link $L_i(i,r)$, CRRD adopts an iterative matching in $IM(i)$. An IM has m output link arbiters, each of which is associated with each output link, and each VOQ has a VOQ arbiter as shown in Figure 3. The switch model that we consider is the same as the one described in Section II-A. However, to simplify the explanation of CRRD, a modification is needed as follows. The order of $VOQ(i,j,h)$ in $IM(i)$ is rearranged for dispatching as $VOQ(i,0,0)$, $VOQ(i,1,0)$, $\cdots$, $VOQ(i,k-1,0)$, $VOQ(i,0,1)$, $VOQ(i,1,1)$, $\cdots$, $VOQ(i,k-1,1)$, $\cdots$, $VOQ(i,0,n-1)$, $VOQ(i,1,n-1)$, $\cdots$, $VOQ(i,k-1,n-1)$. Therefore, $VOQ(i,j,h)$ is redefined as $VOQ(i,v)$, where $v = hk + j$ and $0 \le v \le nk - 1$.

As described in Section II-B, two phases are also considered in CRRD. In phase 1, CRRD employs an iterative matching. The VOQs are selected as candidates and the selected VOQ is assigned to an IM output link. A request that is associated with this output link is sent from the IM to the CM. In phase 2, each selected VOQ that is associated with each output link sends a request from the IM to the CM. The CMs send the arbitration results to the IMs to perform the matching between the IM and the CM.

- Phase 1: Matching within IM
  - First iteration
    - Step 1: Each non-empty VOQ sends a request to every output-link arbiter, each of which is associated with $L_i(i,r)$, where $0 \le i \le k-1$ and $0 \le r \le m-1$.
    - Step 2: Each output link $L_i(i,r)$, where $0 \le r \le m-1$, searches a request among nk non-empty VOQs independently. Each output-link arbiter associated with $L_i(i,r)$ has its own pointer $P_L(i,r)$. The output-link arbiter starts to search

one non-empty VOQ request from the position of $P_L(i,r)$ in a round-robin fashion. Each output-link arbiter sends a grant to a requesting VOQ. Each VOQ has its own round-robin arbiter and one pointer $P_v(i,v)$, where $0 \leq v \leq nk-1$, to choose one output link. The VOQ arbiter starts to search one grant out of several grants that are given by the output-link arbiters from the position of $P_v(i,v)$.

  - ∗ Step 3: The VOQ that chooses one $L_i(i,r)$ by using the round-robin arbiter sends the grant to the selected output link. Note that $P_L(i,r)$ and $P_v(i,v)$ are updated to one position after the granted position, only if they are matched and their requests are also granted by CM in phase 2.
- – ith iteration $(i > 1)$[2]
  - ∗ Step 1: Each unmatched VOQ at the previous iterations sends a request to all the output-link arbiters again.
  - ∗ Steps 2 and 3: The same procedures as in the first iteration are performed.

• Phase 2: Matching between IM and CM

- – Step 1: After the phase 1 is completed, $L_i(i,r)$ sends the request to $CM(r)$. Then arbitration in the CM is performed. Each $CM(r)$ has k pointers $P_c(r,j)$, each of which corresponds to each $OM(j)$. The CM makes its arbitration using the pointer $P_c(r,j)$ in a round-robin fashion, and sends the grants to $L_i(i,r)$ of $IM(i)$. The pointer $P_c(r,j)$ is updated when the CM sends the grant to the IM.
- – Step 2: If the IM receives a grant from the CM, it sends the corresponding cell from that VOQ at the next time slot. Otherwise, the IM cannot send a cell at the next time slot. The request that is not granted by the CM will be dispatched again at the next time slot because the pointers that are related with the ungranted requests are not updated.

Figure 3 shows an example of $n = m = k = 3$, where CRRD is operated at the first iteration in phase 1. At step 1, $VOQ(i,0)$, $VOQ(i,3)$, $VOQ(i,4)$, and $VOQ(i,6)$, which are non-empty VOQs, send requests to all the output-link arbiters. At step 2, output-link arbiters that are associated with $L_i(i,0)$, $L_i(i,1)$, and $L_i(i,2)$ select $VOQ(i,0)$, $VOQ(i,0)$, and $VOQ(i,4)$, respectively, according to their pointers' positions. At step 3, $VOQ(i,0)$ that receives two grants from both output-link arbiters of $L_i(i,0)$ and $L_i(i,1)$, selects $L_i(i,0)$ by using its own VOQ arbiter and sends a grant to the output-link arbiter of $L_i(i,0)$. Since $VOQ(i,1)$ receives one grant from an output-link arbiter $L_i(i,2)$, it sends a grant to the output-link arbiter. At this iteration, $L_i(i,1)$ is not matched with any non-empty VOQs. At the next iteration, the matching between unmatched non-empty VOQs and $L_i(i,1)$ will be performed.

B. Desynchronization Effect

While RD suffers contention at CM, CRRD decreases the contention at the CM because pointers $P_v(i,v)$, $P_L(i,r)$ and $P_c(r,j)$, are desynchronized.

[2]The number of iterations is designed by considering the limitation of the arbitration time in advance. CRRD tries to choose as many non-empty VOQs in this matching as possible, but it does not always choose the maximum available non-empty VOQs if the number of iterations is not enough. On the other hand, RD can choose the maximum number of non-empty VOQs in phase 1.

	T	0	1	2	3	4	5	6	7
$IM(0)$	$P_V(0,0)$	0	1	0	0	0	1	0	0
	$P_V(0,1)$	0	0	1	0	0	0	1	0
	$P_V(0,2)$	0	0	0	1	0	0	0	1
	$P_V(0,3)$	0	0	0	0	1	0	0	0
$IM(1)$	$P_V(1,0)$	0	0	1	0	0	0	1	0
	$P_V(1,1)$	0	0	0	1	0	0	0	1
	$P_V(1,2)$	0	0	0	0	1	0	0	0
	$P_V(1,3)$	0	0	0	0	0	1	0	0
$IM(0)$	$P_L(0,0)$	0	1	2	3	0	1	2	3
	$P_L(0,1)$	0	0	1	2	3	0	1	2
$IM(1)$	$P_L(1,0)$	0	0	1	2	3	0	1	2
	$P_L(1,1)$	0	0	0	1	2	3	0	1
$CM(0)$	$P_C(0,0)$	0	1	0	1	0	1	0	1
	$P_C(0,1)$	0	0	1	0	1	0	1	0
$CM(1)$	$P_C(1,0)$	0	0	1	0	1	0	1	0
	$P_C(1,1)$	0	0	0	1	0	1	0	1

The request is granted by CM.

Fig. 4. Example of desynchronization effect $(n = m = k = 2)$

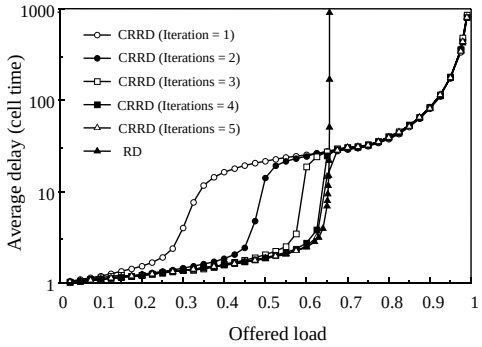

Fig. 5. Delay performance of CRRD and RD schemes $(n = m = k = 8)$

We demonstrate how the pointers are desynchronized by using simple examples. Let us consider the example of $n = m = k = 2$ as shown in Figure 4. We assume that every VOQ is always occupied with cells. Each VOQ sends a request to be selected as a candidate at every time slot. All the pointers are set to $P_v(i,v) = 0$, $P_L(i,r) = 0$, and $P_c(r,j) = 0$ at the initial state. Only one iteration in phase 1 is considered here.

At time slot $T = 0$, since all the pointers are set to 0, only one VOQ in $IM(0)$, which is $VOQ(0,0,0)$, can send a cell through $L_i(0,0)$ to CM(0). The related pointers with the grant, $P_v(0,0)$, $P_L(0,0)$, and $P_c(0,0)$ are updated from 0 to 1. At $T = 1$, three VOQs, which are $VOQ(0,0,0)$, $VOQ(0,1,0)$, and $VOQ(1,0,0)$, can send cells. The related pointers with the grants are updated. Four VOQs can send cells at $T = 2$. In this situation, 100% switch throughput is achieved. There is no contention at all at the CMs from $T = 2$ because the pointers are desynchronized.

We also confirmed that CRRD can achieve the desynchronization effect and provide high-throughput even though the switch size is increased.

IV. Performance of CRRD Scheme

Figure 5 shows that CRRD provides a higher throughput than RD under uniform traffic. A Bernoulli arrival process is used for the input traffic. CRRD can achieve 100% throughput, which is independent of the number of iterations in the IM.

The reason CRRD provides 100% throughput under uni-

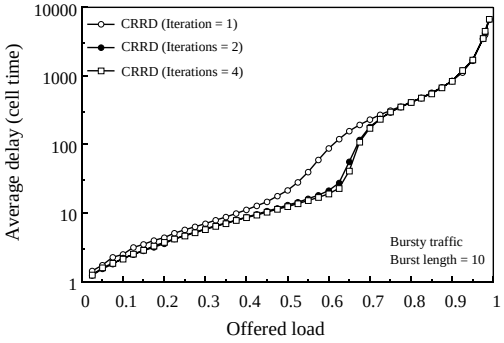

Fig. 6. Delay performance of CRRD affected by bursty traffic ($n = m = k = 8$)

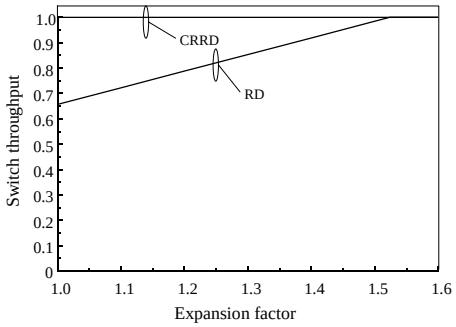

Fig. 7. Relationship between switch throughput and expansion factor ($n = k = 8$)

form traffic is explained as follows. When the offered load is 1.0, and if the idle state, in which the internal link is not fully utilized, still occurs due to the contention in the IM and the CM, a VOQ that fails in the contention has to store backlogged cells. Under uniform traffic, every VOQ keeps backlogged cells until the idle state is eliminated; in other words, until the stable state is reached. The stable state is defined in [8]. In the stable state, every VOQ is occupied with backlogged cells. In this situation, as illustrated in Figures 4, the desynchronization effect is always obtained. Therefore, even when the offered load is 1.0, no contention occurs in the stable state. That is why CRRD provides 100% throughput under uniform traffic, independent of the number of iterations.

As the number of iterations increases, the delay performance improves when the offered load is less than 0.7, as shown in Figure 5. This is because the matching between VOQs and output links $L_i(i, r)$ within the IM increases. When the offered traffic load is not heavy, the desynchronization of the pointers is not completely achieved. At the low offered load, the delay performance of RD is better than that of CRRD with one iteration. This is because the matching within the IM in CRRD is not completely achieved while the complete matching within the IM in RD is always achieved as described in Section II-B. When the offered load is larger than 0.7, the delay performance of CRRD is not improved by increasing the number of iterations in the IM. The number of iterations in the IM improves only the matching within the IM, but does not improve the matching between the IM and CM. The delay performance is improved by the number of iterations in the IM when the offered load is not heavy. In Figure 5, we can see that, when the number of iterations in the IM increases to 4, the delay performance is almost converged.

Figure 6 shows that even when the input traffic is bursty, CRRD provides 100% throughput, which is also independent of the number of iterations, although the delay performance of the bursty traffic becomes worse than that of the non-bursty traffic at the heavy load condition. The reason for the 100% throughput even in the bursty traffic can be explained in the same way as described in the Bernoulli traffic above. We assume that the burst length is exponentially distributed as the bursty traffic. In this evaluation, the burst length is set to be 10.

Figure 7 shows that RD requires the expansion ratio of more than 1.5 to achieve 100% throughput, while CRRD does not need the bandwidth expansion by using simple round-robin arbiters. This is an advantage of CRRD.

V. Conclusions

A Clos-network switch architecture is attractive because of its scalability. The previously proposed implementable dispatching schemes are not able to achieve a high throughput unless the internal bandwidth is expanded.

We have introduced a novel dispatching scheme, CRRD, for the Clos-network switch. CRRD provides high switch throughput without increasing internal bandwidth, while the implementation is very simple because only round-robin arbiters are employed. We confirmed that CRRD achieves 100% throughput under uniform traffic. Even though the offered load reaches 1.0, the pointers of round-robin arbiters that we adopt at the input modules and the central modules are effectively desynchronized and the contention is avoided.

References

[1] T. Chaney, J. A. Fingerhut, M. Flucke, and J. S. Turner, "Design of a Gigabit ATM switch," Proc. IEEE INFOCOM'97, pp. 2-11, Apr. 1997.

[2] H. J. Chao, B-S. Choe, J-S Park, and N. Uzun, "Design and Implementation of Abacus Switch: A Scalable Multicast ATM Switch, " IEEE J. Select. Areas Commun. vol. 15, no. 5, pp. 830-843, 1997.

[3] H. J. Chao and J-S Park, "Centralized Contention Resolution Schemes for a Large-Capacity Optical ATM Switch," Proc. IEEE ATM Workshop'97, Fairfax, VA, May 1998.

[4] H. J. Chao, "Saturn: A Terabit Packet Switch Using Dual Round-Robin," IEEE Commun. Mag., vol. 38, no. 12 pp. 78-84, Dec. 2000.

[5] C. Clos, "A Study of Non-Blocking Switching Networks," Bell Sys. Tech. Jour., pp. 406-424, March 1953.

[6] F. M. Chiussi, J. G. Kneuer, and V. P. Kumar, "Low-Cost Scalable Switching Solutions for Broadband Networking: The ATLANTA Architecture and Chipset," IEEE Commun. Mag. pp. 44-53, Dec. 1997.

[7] N. Mckeown, "Scheduling Algorithm for Input-Queued Cell Switches," Ph. D. Thesis., University of California at Berkeley, 1995.

[8] N. Mckeown, V. Anantharam, and J. Walrand, "Achieving 100% Throughput in an Input-Queued Switch," Proc. IEEE INFOCOM96, pp. 296-302, 1996.

[9] N. Mckeown, M. Izzard, A. Mekkittikul, W. Ellerisick, and M. Horowitz, "Tiny-Tera: A Packet Switch Core," IEEE Micro, pp. 26-33, Jan-Feb. 1997.

[10] N. Mckeown, "The *i*SLIP Scheduling Algorithm for Input-Queues Switches," IEEE Trans. Networking vol. 7, no. 2, pp. 188-200, April, 1999.

[11] E. Oki, N. Yamanaka, Y. Ohtomo, K. Okazaki, and R. Kawano, "A 10-Gb/s (1.25 Gb/s × 8) 4 × 2 0.25-μm CMOS/SIMOX ATM Switch Based on Scalable Distributed Arbitration," IEEE J. Solid-State Circuits, vol. 34, no. 12. pp.1921-1934, Dec. 1999.

[12] J. Turner and N. Yamanaka, "Architectural Choices in Large Scale ATM Switches," IEICE Trans. Commun. vol. E81-B, no. 2, pp. 120-137, Feb. 1998.