PCRRD: A Pipeline-Based Concurrent Round-Robin Dispatching Scheme for Clos-Network Switches

Eiji Oki, Roberto Rojas-Cessa, and H. Jonathan Chao

Abstract—**This paper proposes a pipeline-based concurrent round-robin dispatching scheme, called PCRRD, for Clos-network switches. Our previously proposed concurrent round-robin dispatching (CRRD) scheme provides 100% throughput under uniform traffic by using simple round-robin arbiters, but it has the strict timing constraint that the dispatching scheduling has to be completed within one cell time slot. This is a bottleneck in building high-performance switching systems. To relax the strict timing constraint of CRRD, we propose to use more than one scheduler engine, up to P, so called subschedulers. Each subscheduler is allowed to take more than one time slot for dispatching. Every time slot, one out of P subschedulers provides the dispatching result. The subschedulers adopt our original CRRD algorithm. We show that PCRRD preserves 100% throughput under uniform traffic of our original CRRD algorithm, while ensuring the cell-sequence order. Since the constraint of the scheduling timing is dramatically relaxed, it is suitable for high-performance switching systems even when the switch size increases and a port speed is high (e.g., 40 Gbit/s).**

Keywords— **Packet switch, Clos-network switch, dispatching, arbitration, pipeline, throughput**

I. INTRODUCTION

The explosion of Internet traffic has led to a greater need for high-speed switches and routers that have over 1-Tbit/s throughput. To meet this demand, several high-speed packet switching systems have been described [1], [2]. Most high-speed packet switching systems adopt a fixed-sized cell in the switch fabric. Variable-length packets are segmented into several fixed-sized cells when they arrive, are switched through the switch fabric, and are 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 the single-stage switch architectures is a crossbar switch. There are many studies on developing crossbar-based high-speed switches [1], [3], [4], [5], [6]. Although this crossbar-based approach is effective up to a certain switch size, the number of the switching elements is proportional to the square of the number of switch ports. This makes a large-scale switch difficult to implement.

The other approach is to use a multiple-stage switch architecture, such as a Clos-network switch [7]. The Clos-network switch architecture, which involves 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.

A gigabit ATM (Asynchronous Transfer Mode) switch using buffers in the second-stage was presented in [8]. 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. Implementing buffers in the second-stage modules resolves contention among cells from different first-stage modules [9]. The internal speed-up factor was suggested to be set more than 1.25 in [8]. However, it requires a re-sequence function at the third-stage modules, or the last 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.

This research is supported by NSF Grant 9906673.

Eiji Oki is with NTT Network Innovation Laboratories, NTT Corporation, 3-9-11 Midori-cho Musashino-shi, Tokyo 180-8585 Japan. This work was done while he was a Visiting Scholar at Polytechnic University, Brooklyn, NY. E-mail: oki.eiji@lab.ntt.co.jp

Roberto Rojas-Cessa is with Polytechnic University, 6 Metrotech Center, Brooklyn, NY 11201 USA. He was a CONACYT fellow. Email: rrojas@ieee.org

H. Jonathan Chao is with Polytechnic University, 6 Metrotech Center, Brooklyn, NY 11201 USA. Email: chao@poly.edu

In [10], an ATM switch using non-buffered second-stage modules was developed. This approach does not suffer from the out-of-sequence problem. Since there is no buffer in the second-stage modules to resolve contention, dispatching cells from the first stage to the second stage becomes an important issue. There are some buffers to resolve contention at the first and third stages. A random dispatching (RD) scheme is used for cell dispatching from the first stage to the second stage [10], as adopted in the case of the buffered second-stage modules in [8]. 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. In [11], we derived the relationship between the internal expansion ratio and switch size when the RD scheme is employed. To achieve 100% throughput by using RD, the internal expansion ratio is set to about 1.6 when the switch size is large [10]. This makes a high-speed switch difficult to implement in a cost-effective manner.

It is a challenge to find a cost-effective dispatching scheme that is able to achieve a high throughput in Clos-network switches, without allocating any buffers in the second stage to avoid the out-of-sequence problem and without expanding internal bandwidth.

We introduced a solution to the challenge in [11], where a round-robin-based dispatching scheme, called the concurrent round-robin dispatching (CRRD) scheme, was proposed for a Clos-network switch. The basic idea of CRRD is to use the desynchronization effect [12] in the Clos-network switch. The desynchronization effect has been studied using simple scheduling algorithms as iSLIP [12], [13] and Dual Round-Robin Matching (DRRM) [3], [14] in an input-queued crossbar switch. CRRD provides high switch throughput without expanding the internal bandwidth, while the implementation is simple because only simple round-robin arbiters are employed. We showed that CRRD achieves 100% throughput under uniform traffic.

However, CRRD has a constraint in terms of dispatching scheduling time. The dispatching scheduling has to be completed within one cell time slot. The constraint is a bottleneck when the switch size increases or a port speed becomes fast. This is because the time needed for arbitration becomes longer than one time slot.

Therefore, we need a dispatching scheme that meets the following requirements for practical high-performance switching systems.

- *Dispatching timing relaxation:* The dispatching time should be relaxed into more than one time slot to support the large switch size and a fast port speed.
- *High throughput*
- *Cost-effective implementation:* No queuing buffer in the second stage is allocated to avoid the out-of-sequence problem. In addition, the internal bandwidth is not expanded for a cost-effective implementation.

This paper presents a solution to these requirements. We propose a pipeline-based concurrent round-robin dispatching scheme, called PCRRD, for Clos-network switches. To relax the strict timing constraint of CRRD, we propose to use more than one scheduler engine, up to P, so called subschedulers. Each subscheduler is allowed to take more than one time slot for dispatching. One of them provides the dispatching result every time slot. The subschedulers can adopt our original CRRD algorithm. We show that PCRRD provides 100% throughput under uniform traffic, while preserving the properties of CRRD.

The remainder of this paper is organized as follows. Section II describes a Clos-network switch model that we reference throughout this paper. Section III describes our previously proposed CRRD scheme. Section IV introduces the PCRRD scheme to relax the dispatching timing constraint of CRRD. Section V describes the performance of the PCRRD scheme. Section VI summarizes the key points.

II. Clos-Network Switch model

Figure 1 shows a three-stage Clos-network switch. The terminology used in this paper is 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 $0 \leq i \leq k-1$.
j	OM number, where $0 \leq j \leq k-1$.
h	Input-port (IP)/output-port (OP) number in each IM/OM, respectively, where $0 \leq h \leq n-1$.
r	Central-module (CM) number, where $0 \leq r \leq m-1$.
$IM(i)$	$i+1$th IM.
$CM(r)$	$r+1$th CM.
$OM(j)$	$j+1$th OM.
$IP(i,h)$	$h+1$th input port at $IM(i)$.
$OP(j,h)$	$h+1$th output port at $OM(j)$.
$VOQ(i,v)$	Virtual output queue (VOQ) at $IM(i)$ that stores cells destined for $OP(j,h)$, where $v = hk+j$ and $0 \leq v \leq nk-1$.
$A_V(i,v)$	VOQ arbiter for $VOQ(i,v)$
$L_I(i,r)$	Output link at $IM(i)$ that is connected to $CM(r)$.
$A_L(i,r)$	IM output-link arbiter for $L_I(i,r)$
$L_C(r,j)$	Output link at $CM(r)$ that is connected to $OM(j)$.
$A_C(r,j)$	CM output-link arbiter for $L_C(r,j)$

The first stage consists of k IMs, each of which has an $n \times m$ dimension. The second stage consists of m buffer-less CMs, each of which has a $k \times k$ dimension. The third stage consists of k OMs, each of which has an $m \times n$ dimension.

An $IM(i)$ has nk Virtual Output Queues (VOQs) to eliminate Head-Of-Line (HOL) blocking [10]. A VOQ is denoted as $VOQ(i,v)$. Each $VOQ(i,v)$ stores cells that go from $IM(i)$ to the Output Port $OP(j,h)$ at $OM(j)$. A VOQ can receive, at most, n cells from n input ports in each cell time slot. The HOL cell in each VOQ can be selected for transmission across the switch through $CM(r)$ in each time slot. This ensures that cells are transmitted from the same VOQ in sequence.

Each $IM(i)$ has m output links. An output link $L_I(i,r)$, is connected to each $CM(r)$.

A $CM(r)$ has k output links, each of which is denoted as $L_C(r,j)$, and it is connected to k OMs, each of which is $OM(j)$.

An $OM(j)$ has n output ports, each of which is $OP(j,h)$ and 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.

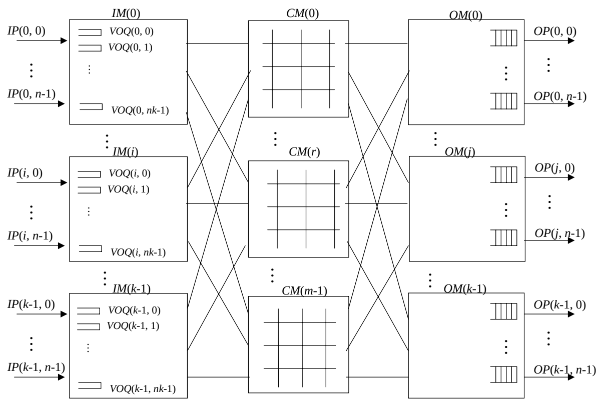


Fig. 1. Clos-network switch with virtual output queues (VOQs) in the input modules

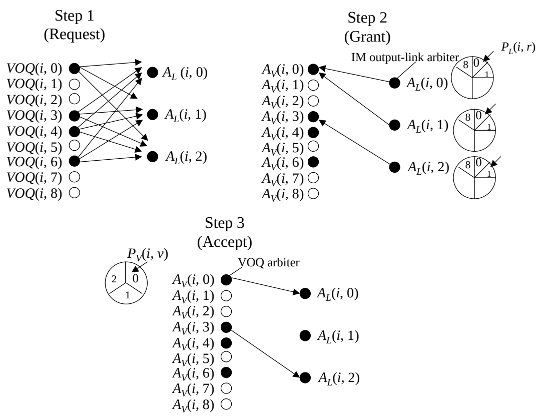


Fig. 2. Concurrent round-robin dispatching (CRRD) scheme

III. Concurrent Round-Robin Dispatching (CRRD) Scheme

To overcome the switch throughput limitation of RD, we developed CRRD, described in [11].

A. CRRD Algorithm

Figure 2 illustrates the detailed CRRD algorithm, which can be considered as the special case when the number of subschedulers is one in PCRRD. To determine the matching between a request from $VOQ(i,v)$ and the output link $L_I(i,r)$, CRRD adopts an iterative matching within $IM(i)$. An IM has m output link arbiters $A_L(i,r)$, each of which is associated with each output link, and each VOQ has a VOQ arbiter $A_V(i,v)$ as shown in Figure 2. In $CM(r)$, there are k round-robin arbiters $A_C(r,j)$, each of which corresponds to $OM(j)$. $A_L(i,r)$, $A_V(i,v)$, and $A_C(r,j)$ have their own round-robin pointers, $P_L(i,r)$, $P_V(i,v)$, and $P_C(r,j)$, respectively.

[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.

We consider two phases for dispatching from the first stage to the second stage. In phase 1, CRRD employs an iterative matching by using round-robin arbiters to assign a VOQ to an IM output link. The matching between VOQs and output links is performed within the IM. It is based on round-robin arbitration and is similar to the iSLIP request/grant/accept approach. A major difference is that in CRRD, a VOQ sends a request to every output-link arbiter; in iSLIP, a VOQ sends a request to only the destined output-link arbiter. In phase 2, each selected VOQ that is matched with an output link sends a request from the IM to the CM. CMs send the arbitration results to IMs to complete the matching between the IM and the CM.

- Phase 1: Matching within IM
  - First iteration
    - ∗ Step 1: Each non-empty $VOQ(i, v)$ sends a request to $A_L(i, r)$ at $L_I(i, r)$.
    - ∗ Step 2 : Each $A_L(i, r)$ chooses one non-empty VOQ request in a round-robin fashion by searching from the position of $P_L(i, r)$. It then sends the grant to the selected VOQ.
    - ∗ Step 3 : The $A_V(i, v)$ sends the accept to the granting $A_L(i, r)$ among all those received in a round-robin fashion by searching from the position of $P_V(i, v)$.
  - ith iteration ($i > 1$)
    - ∗ Step 1 : Each unmatched $VOQ(i, v)$ at the previous iterations sends another request to all unmatched $A_L(i, r)$ again.
    - ∗ Steps 2 and 3 : The same procedure is performed as in the first iteration for matching between unmatched non-empty VOQs and unmatched output links.
- Phase 2: Matching between IM and CM
  - Step 1: After phase 1 is completed, $A_V(i, v)$ sends the request to $CM(r)$. Then, $A_C(r, j)$ chooses one request by searching from the position of $P_C(r, j)$, and sends the grant to $A_V(i, v)$ of $IM(i)$.
  - Step 2: If the IM receives the grant from the CM, it sends a corresponding cell from that VOQ at the next time slot.

As with iSLIP, the round-robin pointers $P_L(i, r)$ and $P_V(i, v)$ in $IM(i)$ and $P_C(r, j)$ in $CM(r)$ are updated to one position after the granted position, only if the matching within the IM is achieved at the first iteration in phase 1 and the request is also granted by the CM in phase 2.

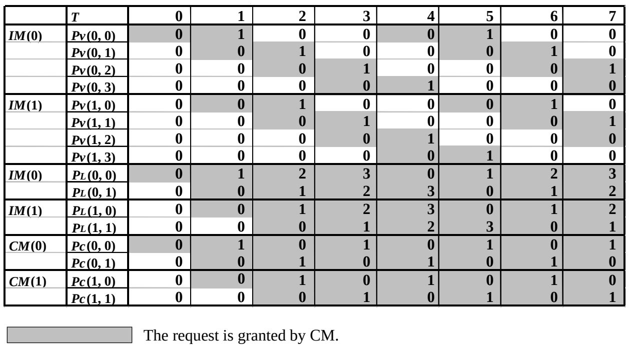
The CRRD algorithm has to be completed within one time slot to provide a matching result every time slot.

Figure 2 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 $A_L(i, r)$. At step 2, $A_L(i, 0)$, $A_L(i, 1)$, and $A_L(i, 2)$, select $VOQ(i, 0)$, $VOQ(i, 0)$, and $VOQ(i, 3)$, respectively, according to their pointers' positions. At step 3, after $A_V(i, 0)$ receives two grants from both $A_L(i, 0)$ and $A_L(i, 1)$, it selects $L_I(i, 0)$. Then it sends a grant to $A_L(i, 0)$. Since $VOQ(i, 3)$ receives one grant from the $A_L(i, 2)$, it sends a grant to it. With one iteration, $L_I(i, 1)$ cannot be matched with any non-empty VOQs. At the next iteration, the matching between unmatched non-empty VOQs and $L_I(i, 1)$ is performed.

B. Desynchronization Effect of CRRD

While RD suffers contention at CM [10], CRRD decreases the contention at the CM because pointers $P_V(i, v)$, $P_L(i, r)$, and $P_C(r, j)$, are desynchronized.

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 3. 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 be $P_V(i, v) = 0$, $P_L(i, r) = 0$, and $P_C(r, j) = 0$ at the initial state. Only one iteration in phase 1 is considered here.



	T	0	1	2	3	4	5	6	7
IM(0)	*Pv*(0, 0)	0	1	0	0	0	1	0	0
	Pv(0, 1)	0	0	1	0	0	0	1	0
	Pv(0, 2)	0	0	0	1	0	0	0	1
	Pv(0, 3)	0	0	0	0	1	0	0	0
IM(1)	*Pv*(1, 0)	0	0	1	0	0	0	1	0
	Pv(1, 1)	0	0	0	1	0	0	0	1
	Pv(1, 2)	0	0	0	0	1	0	0	0
	Pv(1, 3)	0	0	0	0	0	1	0	0
IM(0)	*PL*(0, 0)	0	1	2	3	0	1	2	3
	PL(0, 1)	0	0	1	2	3	0	1	2
IM(1)	*PL*(1, 0)	0	0	1	2	3	0	1	2
	PL(1, 1)	0	0	0	1	2	3	0	1
CM(0)	*Pc*(0, 0)	0	1	0	1	0	1	0	1
	Pc(0, 1)	0	0	1	0	1	0	1	0
CM(1)	*Pc*(1, 0)	0	0	1	0	1	0	1	0
	Pc(1, 1)	0	0	0	1	0	1	0	1

▨ The request is granted by CM.

Fig. 3. Example of desynchronization effect ($n = m = k = 2$)

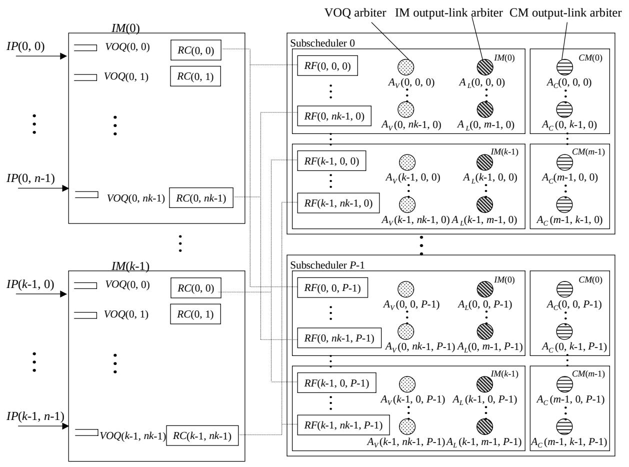


Fig. 4. PCRRD structure (centralized implementation)

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 with $L_I(0, 0)$ through 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 the CMs from $T = 2$ because the pointers are desynchronized.

In the same way as the above example, CRRD can achieve the desynchronization effect and provide high-throughput even though the switch size is increased.

IV. Pipeline-Based Concurrent Round-Robin Dispatching (PCRRD) Scheme

The PCRRD scheme is able to relax the computation time for dispatching into more than one time slot. PCRRD is performed by using nk^2 request counters and P subschedulers for a Clos-network switching system.

When the schedulers are implemented, there are two approaches: centralized and non-centralized. Figures 4 and 5 show examples of centralized and non-centralized approaches, respectively. In the centralized approach, each subscheduler is con-

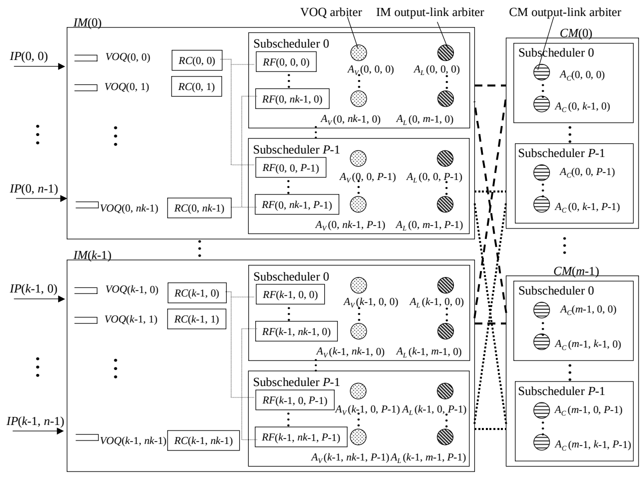


Fig. 5. PCRRD structure (non-centralized implementation)

nected to all IMs, as shown in Figure 4.[2] In the non-centralized approach, the subschedulers are implemented in different locations (i.e., in IMs and CMs as the algorithm description), as shown in Figure 5. Each subscheduler has $k + m$ different parts.

PCRRD is applied in whichever approach is employed. The following explanation of PCRRD is valid for both approaches. In Figures 4 and 5, each subscheduler has three kinds of round-robin arbiters: VOQ arbiter $A_V(i, v, p)$, IM output-link arbiter $A_L(i, r, p)$, and CM output-link arbiter $A_C(r, j, p)$, where $0 \leq p \leq P - 1$.[3] Each subscheduler operates the original CRRD algorithm in a pipelined manner, as shown in Figure 6. Each scheduler takes P time slots to complete the dispatching.

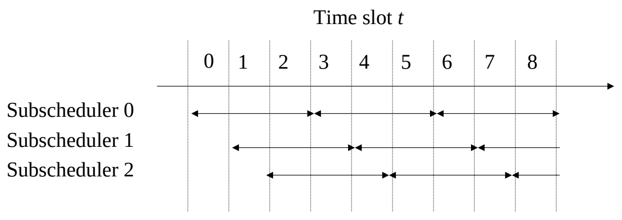


Fig. 6. Timing diagram of PCRRD with $P = 3$

We define several notations used here in the following. A request counter $RC(i, v)$ is associated with $VOQ(i, v)$, as shown in Figures 4 and 5. The value of $RC(i, v)$ is denoted as $C(i, v)$, where $0 \leq C(i, v) \leq L_{max}$. L_{max} is the maximum VOQ occupancy. $C(i, v)$ expresses the number of accumulated requests associated with $VOQ(i, v)$ that have not been sent to any of the subschedulers. Each subscheduler has nk^2 request flags. Each request flag $RF(i, v, p)$ is associated with $VOQ(i, v)$ and subscheduler p. The value of $RF(i, v, p)$ is denoted as $F(i, v, p)$, where $0 \leq F(i, v, p) \leq 1$. $F(i, v, p) = 1$ means that $VOQ(i, v)$ has a request in subscheduler p. $F(i, v, p) = 0$ means that $VOQ(i, v)$ has no request in subscheduler p. At initial time, $C(i, v)$ and $F(i, v, p)$ are set to zero.

PCRRD operates as follows.

- Stage 1: When a new cell enters $VOQ(i, v)$, the value of counter $RC(i, v, p)$ is increased: $C(i, v) \leftarrow C(i, v) + 1$.
- Stage 2: At the beginning of every time slot t, if $C(i, v) > 0$ and $F(i, v, p) = 0$, where $p = t \mod P$, $C(i, v) \leftarrow C(i, v) - 1$ and $F(i, v, p) \leftarrow 1$. Otherwise, $C(i, v)$ and $F(i, v, p)$ are not changed.
- Stage 3: At $Pl + p \leq t < P(l + 1) + p$, where l is an integer, subscheduler p operates the original CRRD algorithm.
- Stage 4: By the end of every time slot t, subscheduler p, where $p = (t - (P - 1)) \mod P$, completes a matching. When $RF(i, v, p)$ is granted, $F(i, j, k) \leftarrow F(i, j, k) - 1$. In this case, The HOL cell in $VOQ(i, v)$ is sent to $OP(j, h)$ at $OM(j)$ through a CM at the next time slot.[4] When $RF(i, v, p)$ is not granted, $F(i, v, p)$ is not changed.

Whenever a condition associated with any stage is satisfied, the stage is executed.

To apply the original CRRD algorithm in subscheduler p, we use $F(i, v, p)$ instead of VOQ requests as described in Section III. Each subscheduler has its own round-robin pointers. The pointers in subscheduler p are independent of those in other subschedulers. In Figure 4, subscheduler p does not directly communicate with subscheduler p', where $p \neq p'$. In the same way, Figure 5 shows that subscheduler p in IM does not directly communicate with subscheduler p' in CM, where $p \neq p'$, although subscheduler p in IM communicates with subscheduler p in CM. Parts of subscheduler p in IMs and CMs communicates with each other.

V. Performance of PCRRD

This section describes throughput and delay performance of PCRRD under uniform traffic. In addition, the effect of the PCRRD scheduling relaxation is described.

A. Throughput

PCRRD, which uses the CRRD algorithm in the subschedulers, provides 100% throughput under uniform traffic.

The reason is as follows. Consider the input load as 1.0. If an IM cannot send m cells, outstanding requests are maintained in each subscheduler, in other words, $F(i, v, p) = 1$. As a result, $C(i, v)$ is not always decremented in stage 2 and increased in stage 1. Since $C(i, v)$ reaches a large value enough to be always satisfied with $C(i, v) > 0$, $F(i, v, p) = 1$ is kept in stage 2[5]. In this situation with $F(i, v, p) = 1$ at any t in stage 3, subscheduler p provides a complete matching result every P time slot due to the desynchronization effect of all round-robin arbiters in each subscheduler, as described in Section III. We note that the pointer desynchronization is achieved in the same subscheduler. Pointers of each subscheduler behave independent of others.

Thus, PCRRD preserves the throughput advantage of CRRD.

B. Delay

Figure 7 shows that using P subschedulers, the delay performance of the original algorithm is not affected significantly. Bernoulli arrival process is used for the input traffic. We assume that the switch size N is 64, where $n = m = k = 8$. In this evaluation, we include absolute delay caused by the scheduling time in the delay performance. When P increases, requests from $RC(i, v)$ are distributed among associated subschedulers. Therefore, the desynchronization effect becomes less efficient with P for a light traffic load. For a heavy traffic load, the delay

[2]In the example in Figure 4, each subscheduler is not connected to any CMs. Once a dispatching route is determined, routing bits are attached in the header of the head-of-line cell in a VOQ and the cell is transmitted from the IM.

[3]The arbiters are allocated as in CRRD; however, a new dimension is added, p.

[4]This ensures that cells from the same VOQ are transmitted in sequence, even if $L(i, v) - C(i, v) > 1$, where $L(i, v)$ is the occupancy of $VOQ(i, v)$. Note that $L(i, v) - C(i, v) = \sum_{p=0}^{P-1} F(i, v, p) \leq P$. $L(i, v) - C(i, v)$ is the number of outstanding requests of $VOQ(i, v)$ that are not granted by subschedulers.

[5]Although $F(i, v, p)$ becomes 0 in stage 4 when a request is granted, $F(i, v, p)$ is always changed to 1 in stage 2.

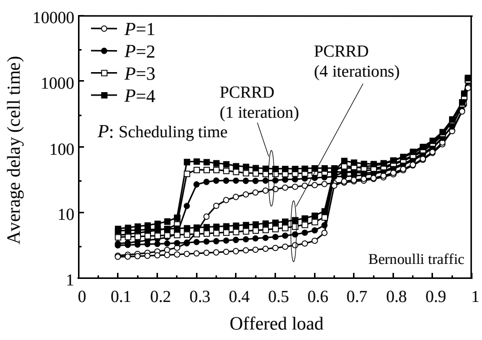


Fig. 7. Delay dependency on scheduling time P for Bernoulli traffic ($n = m = k = 8$)

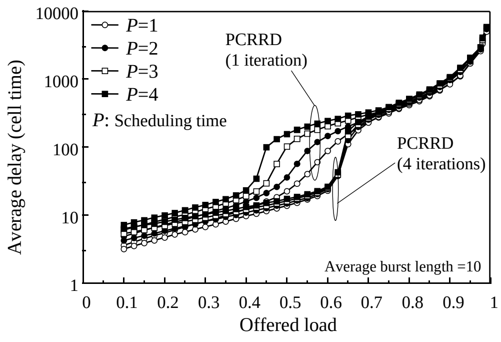


Fig. 8. Delay dependency on scheduling time P for Bursty traffic ($n = m = k = 8$)

dependency on P becomes negligible. Therefore, P does not affect delay performance in a practical use.

The delay performance is improved with more iterations. Since PCRRD relaxes the dispatching timing constraint, a large number of iterations within the IM are able to be adopted even when the switch size increases or a port speed becomes fast, compared with the non-pipelined algorithm. Note that we showed the delay performance up to four iterations because there is no measurable improvement with more iterations [11].

Figure 8 shows that even when input traffic is bursty, P does not affect delay performance for a practical use. We assume that the burst length exponentially is distributed as bursty traffic. The burst length is set to 10.

C. Scheduling Timing Relaxation

Figure 9 shows the effect of the PCRRD scheduling timing relaxation. We assume that a cell size, L_{cell} is 64×8 bits. Let the allowable scheduling time, a port speed, and the number of iterations, be T_{sch}, C, and I. T_{sch} is given by,

$$T_{sch} = \frac{PL_{cell}}{C}. \quad (1)$$

T_{sch} decreases with C, but increases with P. In the non-pipelined CRRD scheme, P is equal to 1 as a special case of PCRRD in Eq. (1). In the non-pipelined CRRD, when C=40 Gbit/s, T_{arb}=12.8 ns. Under this timing constraint, it is difficult to implement round-robin arbiters that support large N by using current available CMOS technologies. On the other hand, PCRRD can expand T_{sch} by increasing P, when C=40 Gbit/s and $P = 4$, T_{arb}=51.2 ns. Therefore, PCRRD expands the allowable scheduling time for dispatching so that it can support the desired port speed even when N increases.

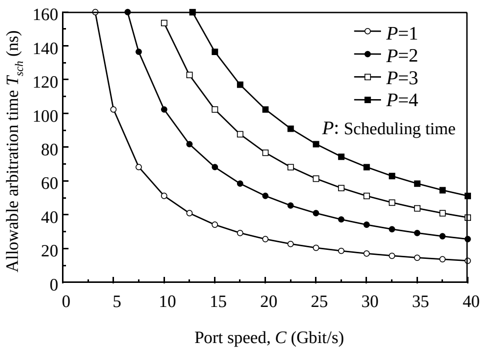


Fig. 9. Effect of PCRRD scheduling timing relaxation

VI. Conclusions

This paper proposed a pipeline-based concurrent round-robin dispatching scheme, called PCRRD, for Clos-network switches. To relax the dispatching timing constraint, we introduced more than one subscheduler, each of which is allowed to take more than one time slot for dispatching. One of them provides the dispatching result every time slot. The subschedulers adopt our original CRRD algorithm. They are implemented in a distributed fashion. We showed that PCRRD preserves 100% throughput under uniform traffic with our original CRRD algorithm, while ensuring that cells from the same VOQ are transmitted in sequence. In addition, we observed that the dispatching scheduling time does not affect delay performance for a practical use. Since the constraint of the dispatching scheduling timing is dramatically relaxed, it is suitable for high-performance switching systems even when the switch size increases and the port speed becomes high.

References

[1] 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.

[2] N. Yamanaka, E. Oki, S. Yasukawa, R. Kawano, and K. Okazaki, "OPTIMA: Scalable, Multi-Stage, 640-Gbit/s ATM Switching System Based on Advanced Electronic and Optical WDM Technologies," IEICE Trans. Commun., vol. E83-B, no. 7,

[3] H. J. Chao and J-S Park, "Centralized Contention Resolution Schemes for a Large-Capacity Optical ATM switch," Proc. IEEE ATM Workshop'98, Fairfax, VA, May 1998.

[4] G. Nong and M. Hamdi, "On the Provision of Quality-of-Service Guarantees for Input Queued Switches," IEEE Commun. Mag., vol. 38, no. 12, Dec. 2000, pp.62-69.

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

[6] R. Rojas-Cessa, E. Oki, Z. Jing, and H. J. Chao, "CIXB-1: Combined Input-One-Cell-Crosspoint Buffered Switch," Proc. 2001 IEEE Workshop on High Performance Switching and Routing (HPSR), 2001.

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

[8] 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.

[9] 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. pp. 1488-1496, 2000.

[10] 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. pp.1921-1934, Dec. 1999.

[11] E. Oki, Z. Jing, R. Rojas-Cessa, and H. J. Chao, "Concurrent Round-Robin Dispatching Scheme in a Clos-Network Switch," Proc. IEEE ICC 2001, pp. 106-112, June 2001.

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

[13] N. McKeown, "The iSLIP Scheduling Algorithm for Input-Queues Switches," IEEE Trans. Networking vol. 7, no. 2, pp. 188-200, April, 1999.

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