Concurrent Round-Robin-Based Dispatching Schemes for Clos-Network Switches*

Eiji Oki[1]†, Zhigang Jing[2], Roberto Rojas-Cessa[2], and H. Jonathan Chao[2]

[1]NTT Network Innovation Laboratories
3-9-11 Midori-cho, Musashino-shi, Tokyo, 180-8585 Japan
email: oki.eiji@lab.ntt.co.jp

[2]Department of Electrical Engineering
Polytechnic University
6 Metrotech Center, Brooklyn, New York 11201 USA
email: {zgjing, rrojas, chao}@kings.poly.edu

February 16, 2002

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 (RD), are not able to achieve high throughput unless the internal bandwidth is expanded. This paper presents two round-robin-based dispatching schemes to overcome the throughput limitation of the RD scheme. First, we introduce a concurrent round-robin dispatching (CRRD) scheme for the Clos-network switch. The CRRD scheme provides high switch throughput without expanding internal bandwidth. CRRD implementation is very simple because only simple round-robin arbiters are adopted. We show via simulation 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 completely desynchronized and contention is avoided. Second, we introduce a concurrent master-slave round-robin dispatching (CMSD) scheme as an improved version of CRRD to make it more scalable. CMSD uses hierarchical round-robin arbitration. We show that CMSD preserves the advantages of CRRD, reduces the scheduling time by 30% or more when arbitration time is significant, and has a dramatically reduced number of crosspoints of the interconnection wires between round-robin arbiters in the dispatching scheduler with a ratio of $\frac{1}{\sqrt{N}}$, where N is the switch size. This makes CMSD easier to implement than CRRD when the switch size becomes large.

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

*This research is supported by NSF Grant 9906673 and ANI-9814856.

†Eiji Oki is a corresponding author. This work was done while he was a Visiting Scholar at Polytechnic University.

1 Introduction

As the Internet continues to grow, high-capacity switches and routers are needed for backbone networks. Several approaches have been presented for high-speed packet switching systems [2, 3, 13]. Most high-speed packet switching systems use a fixed-sized 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 the single-stage architecture is a crossbar switch. Identical switching elements are arranged on a matrix plane. Several high-speed crossbar switches are described [4, 13, 16]. However, the number of I/O pins in a crossbar chip limits the switch size. This makes a large-scale switch difficult to implement cost-effectively as the number of chips becomes large.

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

[2] 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. The purpose of implementing buffers in the second-stage modules is to resolve contention among cells from different first-stage modules [18]. The internal speed-up factor was suggested to be set more than 1.25 in [2]. However, this 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 a switch more difficult to implement.

[6] 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. There are some buffers to absorb contention at the first and third stages.[1]

A random dispatching (RD) scheme is 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 [2]. This scheme attempts to 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 under uniform traffic [2] by using RD, the internal

[1] Although there are several studies on routing algorithms [9, 10] where every stage has no buffer, the assumption used in the literature is out of the scope of this paper. This is because the assumption requires a contention resolution function for output ports, before cells enter the Clos-network switch. This function is not easy to implement in high-speed switching systems.

[2] A switch can achieve 100% throughput under uniform traffic if the switch is stable. The stable sate is defined in [12, 15]. Note that the 100% definition in [15] is more general than the one used here because all independent and

expansion ratio has to be set to about 1.6 when the switch size is large [6].

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 two solutions to this question. First, 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 [11] in the Clos-network switch. The desynchronization effect has been studied using such simple scheduling algorithms as *i*SLIP [11, 14] and Dual Round-Robin Matching (DRRM) [4, 5] in an input-queued crossbar switch. CRRD provides high switch throughput without increasing internal bandwidth, and its implementation is very simple because only simple round-robin arbiters are employed. We show via simulation that CRRD achieves 100% throughput under uniform traffic. With slightly unbalanced traffic, we also show that CRRD provides a better performance than RD.

Second, this paper describes a scalable round-robin-based dispatching scheme, called the concurrent master-slave round-robin dispatching (CMSD) scheme. CMSD is an improved version of CRRD that provides more scalability. To make CRRD more scalable while preserving CRRD's advantages, we introduce two sets of output-link round-robin arbiters in the first-stage module, which are master arbiters and slave arbiters. These two sets of arbiters operate in a hierarchical round-robin manner. The dispatching scheduling time is reduced, as is the interconnection complexity of the dispatching scheduler. This makes the hardware of CMSD easier to implement than that of CRRD when the switch size becomes large. In addition, CMSD preserves the advantage of CRRD where the desynchronization effect is obtained in a Clos-network switch. Simulation suggests that CMSD also achieves 100% throughput under uniform traffic without expanding internal switch capacity.

Figure 1 categorizes several scheduling schemes and shows the analogy of our proposed schemes. In crossbar switches, *i*SLIP, a round-robin-based algorithm, was developed to overcome the throughput limitation of the parallel iterative matching (PIM) algorithm [1] which uses randomness. In Clos-network switches, CRRD and CMSD, two round-robin-based algorithms, are developed to overcome the throughput limitation and the high implementation complexity of RD which also uses randomness.

The remainder of this paper is organized as follows. Section 2 describes a Clos-network switch model that we reference throughout this paper. Section 3 explains throughput limitation of RD. Section 4 introduces CRRD. Section 5 describes CMSD as an improved version of CRRD. Section 6 presents a performance study of CRRD and CMSD. Section 7 discusses implementation of CRRD and CMSD. Section 8 summarizes the key points.

2 Clos-Network Switch model

Figure 2 shows a three-stage Clos-network switch. The terminology used in this paper is as follows.

admissible arrivals are considered in [15].

Switch type	Scheduling scheme	
	Random	Round robin
Crossbar	PIM[1]	*i*SLIP[11] [14]
Clos network	RD[6]	CRRD and CMSD

Figure 1: Analogy among scheduling schemes

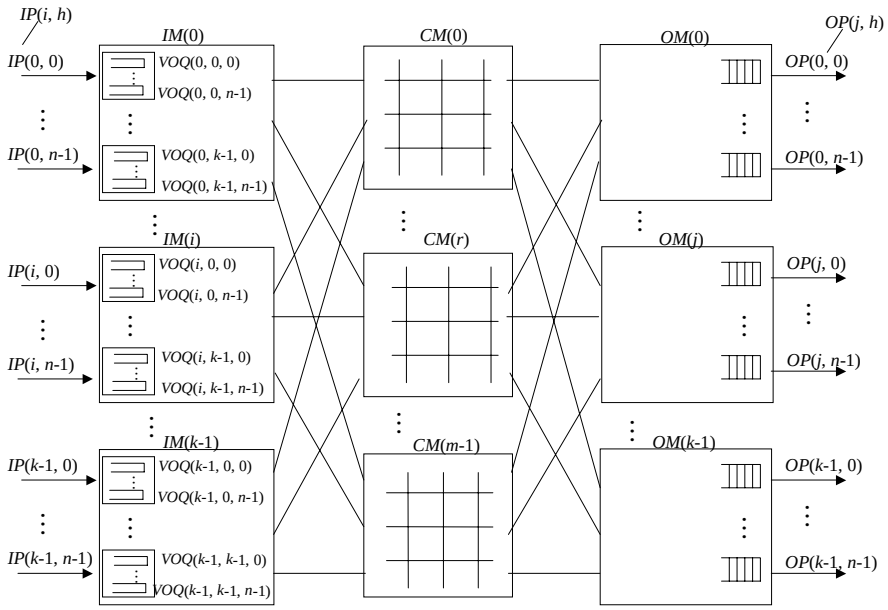


Figure 2: Clos-network switch with virtual output queues (VOQs) in the input modules

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 $OM(j)$.
$VOQ(i,j,h)$	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 $OM(j)$.

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

An $IM(i)$ has nk Virtual Output Queues (VOQs) to eliminate Head-Of-Line (HOL) blocking. A VOQ is denoted as $VOQ(i,j,h)$. Each $VOQ(i,j,h)$ 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 and can send one cell to a CM in one cell time slot.[3] Each $IM(i)$ has m output links. Each 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 $L_C(r,j)$, which are 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.[4] Each output buffer receives at most m cells at one cell time slot, and each output port at the OM forwards one cell in a first-in-first-out (FIFO) manner to the output line.[5]

[3] An L-bit cell must be written to or read from a VOQ memory in a time less than $\frac{L}{C(n+1)}$, where C is a line speed. For example, when $L = 64 \times 8$ bits, C=10 Gbit/s, and $n = 8$, $\frac{L}{C(n+1)}$ is 5.6 ns. This is feasible when we consider current available CMOS technologies.

[4] We assume that the output buffer size at $OP(j,h)$ is large enough to avoid cell loss without flow control between IMs and OMs so that we can focus the discussion on the properties of dispatching schemes in this paper. However, the flow control mechanism can be adopted between IMs and OMs to avoid cell loss when the output buffer size is limited.

[5] Similar to the VOQ memory, an L-bit cell must be written to or read from an output memory in a time less than $\frac{L}{C(m+1)}$.

3 Random Dispatching (RD) Scheme

The ATLANTA switch, developed by Lucent Technologies, uses the random selection in its dispatching algorithm [6]. This section describes the basic concept and performance characteristics of the RD scheme. This description helps to understand the CRRD and CMSD schemes.

3.1 Random Dispatching (RD) Algorithm

Two phases are considered for dispatching from the first stage to the second stage. The details of RD are described in [6]. We show an example of RD in Section 3.2. In phase 1, up to m VOQs from each IM are selected as candidates and a 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. This matching between VOQs and output links is performed only within the 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 the IM to the CM. CMs respond with the arbitration results to IMs so that the matching between IMs and CMs can be completed.

- Phase 1: Matching within IM
  - Step 1: At each time slot, non-empty VOQs send requests for candidate selection.
  - Step 2: $IM(i)$ selects up to 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 up to m candidate cells to *randomly* selected CMs.
- 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 $CM(r)$. A $CM(r)$ has k output links $L_C(r, j)$, each of which corresponds to an $OM(j)$. 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.[6] $CM(r)$ 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 the 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.

3.2 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 3 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.

[6] Even when the round-robin scheme is employed as the contention resolution function for $L_C(r, j)$ at $CM(r)$, the result that we will discuss later is the same as the random selection due to the effect of the random dispatching.

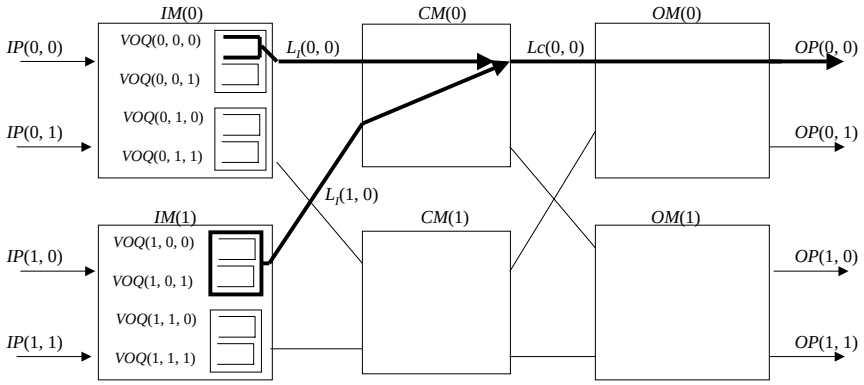


Figure 3: Example of random dispatching (RD) scheme ($n = m = k = 2$)

We estimate how much utilization an output link $OP(j,h)$ can achieve for the cell transmission.[7] Since the utilization of every $OP(i,j)$ is the same, we focus only on 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 $IM(1)$ requests for $L_C(0,0)$ using $L_I(1,0)$. If either $VOQ(1,0,0)$ or $VOQ(1,0,1)$ requests 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.

[7] The output-link utilization is defined as the probability that the output link is used in cell transmission. Since we assume that the traffic load destined for each output port is *not* less than 1.0, we consider the maximum switch throughput as the output-link utilization throughout this paper, unless specifically stated otherwise.

This observation can be generally extended. We derive the formula that gives the maximum throughput T_{max} by using RD under uniform traffic in the following equation.

$$T_{max} = \min\left\{\frac{m}{n}\sum_{i=0}^{k-1}\left(\begin{array}{c} k-1 \\ k-i-1 \end{array}\right)\left(\frac{1}{k}\right)^{k-i-1}\left(1-\frac{1}{k}\right)^{i}\frac{1}{k-i}, 1.0\right\} \quad (3)$$

We describe how to obtain T_{max} in detail in Appendix A. 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 $T_{max} \to 1 - \frac{1}{e} \approx 63\%$ (see Appendix B). In other words, to achieve 100% throughput by using RD, the expansion ratio $\frac{m}{n}$ has to be set to at least $(1 - \frac{1}{e})^{-1} \approx 1.582$.

4 Concurrent Round-Robin Dispatching (CRRD) Scheme

4.1 CRRD Algorithm

The switch model described in this section is the same as the one described in Section 2. However, to simplify the explanation of CRRD, the order of $VOQ(i, j, h)$ in $IM(i)$ is rearranged for dispatching as follows.

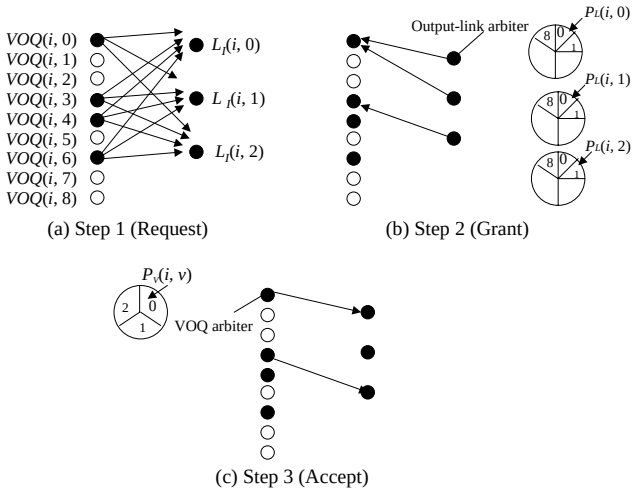


Figure 4: Concurrent round-robin dispatching (CRRD) scheme

$$
\begin{aligned}
&VOQ(i,0,0),\\
&VOQ(i,1,0),\\
&\cdots,\\
&VOQ(i,k-1,0),\\
&VOQ(i,0,1),\\
&VOQ(i,1,1),\\
&\cdots,\\
&VOQ(i,k-1,1)\\
&\cdots,\\
&VOQ(i,0,n-1)\\
&VOQ(i,1,n-1)\\
&\cdots,\\
&VOQ(i,k-1,n-1)
\end{aligned}
$$

Therefore, $VOQ(i,j,h)$ is redefined as $VOQ(i,v)$, where $v = hk + j$ and $0 \leq v \leq nk - 1$.[8]

Figure 4 illustrates the detailed CRRD algorithm by showing an example. To determine the matching between a request from VOQ(i,j,h) and the output link $L_I(i,r)$, CRRD uses an iterative

[8] The purpose of redefining the VOQs is to easily obtain the desynchronization effect.

matching in $IM(i)$. In $IM(i)$, there are m output-link round-robin arbiters and nk VOQ round-robin arbiters. An output-link arbiter associated with $L(i,r)$ and has its own pointer $P_L(i,r)$. A VOQ arbiter associated with $VOQ(i,v)$ has its own pointer $P_V(i,v)$. In $CM(r)$, there are k round-robin arbiters, each of which corresponds to $OM(j)$ and has its own pointer $P_c(r,j)$.

As described in Section 3.1, two phases are also considered in CRRD. 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 *i*SLIP request/grant/accept approach. A major difference is that in CRRD, a VOQ sends a request to every output-link arbiter; in *i*SLIP, 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 sends a request to every output-link arbiter.
    - Step 2 : Each output-link arbiter 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 VOQ arbiter sends the accept to the granting output-link arbiter among all those received in a round-robin fashion by searching from the position of $P_V(i,v)$.
  - ith iteration $(i > 1)$[9]
    - Step 1 : Each unmatched VOQ at the previous iterations sends another request to all unmatched output-link arbiters.
    - 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, $L_I(i,r)$ sends the request to $CM(r)$. Then, each round-robin arbiter associated with $OM(j)$ chooses one request by searching from the position of $P_c(r,j)$, and sends the grant to $L_I(i,r)$ 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. Otherwise, the IM cannot send the cell at the next time slot. The request from the CM that is not granted will be again attempted to be matched at the next time slot because the pointers that are related to the ungranted requests are not moved.

[9] 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.

As with *i*SLIP, 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.

Figure 4 shows an example of $n = m = k = 3$, where CRRD operates 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, $VOQ(i,0)$ receives two grants from both output-link arbiters of $L_I(i,0)$ and $L_I(i,1)$, and accepts $L_I(i,0)$ by using its own VOQ arbiter. Since $VOQ(i,1)$ receives one grant from output-link arbiter $L_I(i,2)$, it accepts the grant. 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 $L_I(i,1)$ will be performed.

4.2 Desynchronization Effect of CRRD

While RD causes contention at CM, as described in Appendix A, CRRD decreases contention at 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 5. 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.

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 all at the CMs from $T = 2$ because the pointers are desynchronized.

Similar to the above example, CRRD can achieve the desynchronization effect and provide high throughput even though the switch size is increased.

5 Concurrent Master-Slave Round-Robin Dispatching (CMSD) Scheme

CRRD overcomes the problem of limited switch throughput of the RD scheme by using simple round-robin arbiters. CMSD is an improved version of CRRD that preserves CRRD's advantages and provides more scalability.

5.1 CMSD Algorithm

Two phases are also considered in the description of CMSD scheme, as in the CRRD scheme. The difference between CMSD and CRRD is how the iterative matching within the IM operates in phase 1.

	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.

Figure 5: Example of desynchronization effect of CRRD ($n = m = k = 2$)

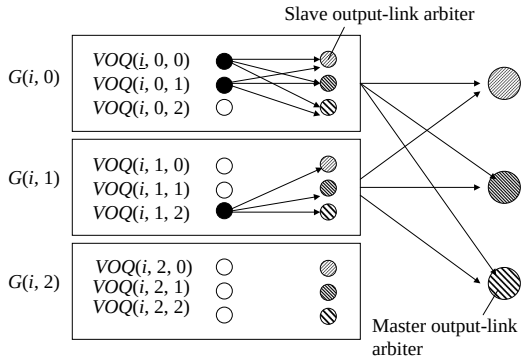
We define several notations to describe the CMSD algorithm, which is shown in an example in Figure 6. $G(i,j)$ is denoted as a VOQ group that consists of n VOQs, each of which is denoted as $VOQ(i,j,h)$. In $IM(i)$, there are m master output-link round-robin arbiters, mk slave output-link round-robin arbiters, and nk VOQ round-robin arbiters. Each master output-link arbiter associated with $L_I(i,r)$ is denoted as $ML(i,r)$ and has its own pointer, $P_{ML}(i,r)$. Each slave output-link arbiter associated with $L_I(i,r)$ and $G(i,j)$ is denoted as $SL(i,j,r)$ and has its own pointer, $P_{SL}(i,j,r)$. A VOQ arbiter associated with $VOQ(i,j,h)$ has its own pointer, $P_V(i,j,h)$.

- Phase 1: Matching within IM
  - First iteration
    - Step 1: There are two sets of requests issued to the output-link arbiters. One set is a request that is sent from a non-empty $VOQ(i,j,h)$ to every associated $SL(i,j,r)$ within $G(i,j)$. The other set is a group-level request sent from $G(i,j)$ that has at least one non-empty VOQ to every $ML(i,r)$.
    - Step 2: Each $ML(i,r)$ chooses a request among k VOQ groups independently in a round-robin fashion by searching from the position of $P_{ML}(i,r)$ and sends the grant to $SL(i,j,r)$ that belongs to the selected $G(i,j)$. $SL(i,j,r)$ receives the grant from its master arbiter and selects one VOQ request in a round-robin fashion by searching from the position of $P_{SL}(i,j,r)$[10] and sends the grant to the selected VOQ.
    - Step 3 : The VOQ arbiter sends the accept to the granting master and slave output-link arbiters among all those received in a round-robin fashion by searching from the position of $P_V(i,j,h)$.
  - ith iteration ($i > 1$)
    - Step 1: Each unmatched VOQ at the previous iterations sends a request to all the slave output-link arbiters again. $G(i,j)$, which has at least one unmatched non-empty VOQ, sends a request to all the unmatched master output-link arbiters again.
    - Steps 2 and 3: The same procedure is performed as in the first iteration for matching between the unmatched non-empty VOQs and unmatched output links.
- Phase 2: Matching between IM and CM
  This operation is the same as in CRRD.

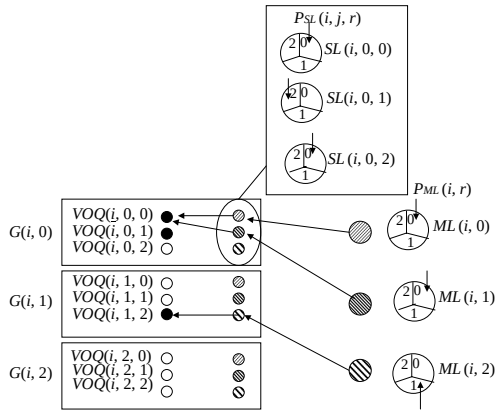
As with CRRD, the round-robin pointers $P_{ML}(i,r)$, $P_{SL}(i,j,r)$, and $P_V(i,j,h)$ in $IM(i)$ and $P_c(r,j)$ in $CM(r)$ are updated to one position after the granted position only if the matching within the IM is achieved at the first iteration in phase 1 and the request is also granted by the CM in phase 2.

Figure 6 shows an example of $n = m = k = 3$, where CMSD is used at the first iteration in phase 1. At step 1, $VOQ(i,0,0)$, $VOQ(i,0,1)$, and $VOQ(i,1,2)$, which are non-empty VOQs, send

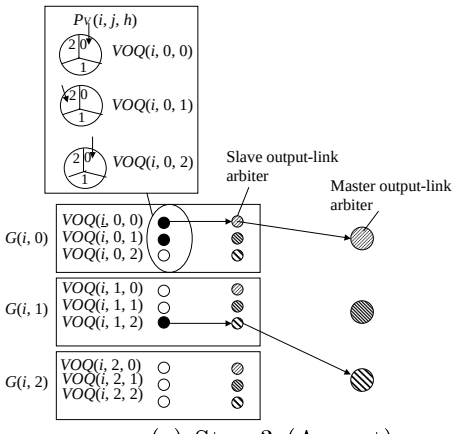
[10] In the actual hardware design, $SL(i,j,r)$ can start to search for a VOQ request without waiting to receive the grant from a master arbiter. If $SL(i,j,r)$ does not receive a grant, the search by $SL(i,j,r)$ is invalid. This is an advantage of CMSD. Section 7.1 discusses the dispatching scheduling time.



(a) Step 1 (Request)



(b) Step 2 (Grant)



(c) Step 3 (Accept)

Figure 6: Concurrent master-slave round-robin dispatching (CMSD) scheme

requests to the associated slave output-link arbiters. $G(i,0)$ and $G(i,1)$ send requests to all the master output-link arbiters. At step 2, $ML(i,0)$, $ML(i,1)$, and $ML(i,2)$ select $G(i,0)$, $G(i,0)$, and $G(i,1)$, respectively, according to the pointers' positions. $SL(i,0,0)$, $SL(i,0,1)$, and $SL(i,1,2)$, which belong to the selected VOQ groups, choose $VOQ(i,0,0)$, $VOQ(i,0,1)$, and $VOQ(i,1,2)$, respectively. At step 3, $VOQ(i,0,0)$ receives two grants from both $SL(i,0,0)$ and $SL(i,0,1)$. It accepts $SL(i,0,0)$ by using its own VOQ arbiter. Since $VOQ(i,1,2)$ receives one grant from $SL(i,1,2)$, it accepts the grant. In this iteration, $L_I(i,1)$ is not matched with any non-empty VOQs. At the next iteration, the matching between an unmatched non-empty VOQ and $L_I(i,1)$ will be performed.

5.2 Desynchronization Effect of CMSD

CMSD also decreases contention at CM because pointers $P_V(i,j,h)$, $P_{ML}(i,r)$, $P_{SL}(i,j,r)$, and $P_c(r,j)$, are desynchronized.

We illustrate how the pointers are desynchronized by using simple examples. Let us consider the example of $n = m = k = 2$ as shown in Figure 7. 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 $P_V(i,j,h) = 0$, $P_{ML}(i,r) = 0$, $P_{SL}(i,j,r) = 0$, 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 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.

Similar to the above example, CMSD can also achieve the desynchronization effect and provide high throughput even though the switch size is increased, as in CRRD.

6 Performance of CRRD and CMSD

The performance of CRRD and CMSD was evaluated by simulation using uniform and non-uniform traffic.

6.1 Uniform Traffic

6.1.1 CRRD

Figure 8 shows that CRRD provides higher throughput than RD under uniform traffic. A Bernoulli arrival process is used for input traffic. Simulation suggests that CRRD can achieve 100% throughput for any number of iterations in IM.

The reason CRRD provides 100% throughput under uniform traffic in simulation 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 contention in IM and CM, a VOQ that fails in the contention has to

		T	0	1	2	3	4	5	6	7
IM(0)	*G*(0, 0)	*Pv*(0, 0, 0)	0	1	0	0	0	1	0	0
		Pv(0, 0, 1)	0	0	0	1	0	0	0	1
	G(0, 1)	*Pv*(0, 1, 0)	0	0	1	0	0	0	1	0
		Pv(0, 1, 1)	0	0	0	0	1	0	0	0
IM(1)	*G*(1, 0)	*Pv*(1, 0, 0)	0	0	1	0	0	0	1	0
		Pv(1, 0, 1)	0	0	0	0	1	0	0	0
	G(1, 1)	*Pv*(1, 1, 0)	0	0	0	1	0	0	0	1
		Pv(1, 1, 1)	0	0	0	0	0	1	0	0
IM(0)	Master	*PML*(0, 0)	0	1	0	1	0	1	0	1
		PML(0, 1)	0	0	1	0	1	0	1	0
	Slave	*PSL*(0, 0, 0)	0	1	1	0	0	1	1	0
	G(0, 0)	*PSL*(0, 0, 1)	0	0	1	1	0	0	1	1
	Slave	*PSL*(0, 1, 0)	0	0	1	1	0	0	1	1
	G(0, 1)	*PSL*(0, 1, 1)	0	0	0	1	1	0	0	1
IM(1)	Master	*PML*(1, 0)	0	0	1	0	1	0	1	0
		PML(1, 1)	0	0	0	1	0	1	0	1
	Slave	*PSL*(1, 0, 0)	0	0	1	1	0	0	1	1
	G(1, 0)	*PSL*(1, 0, 1)	0	0	0	1	1	0	0	1
	Slave	*PSL*(1, 1, 0)	0	0	0	1	1	0	0	1
	G(1, 1)	*PSL*(1, 1, 1)	0	0	0	0	1	1	0	0
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.

Figure 7: Example of desynchronization effect ($n = m = k = 2$)

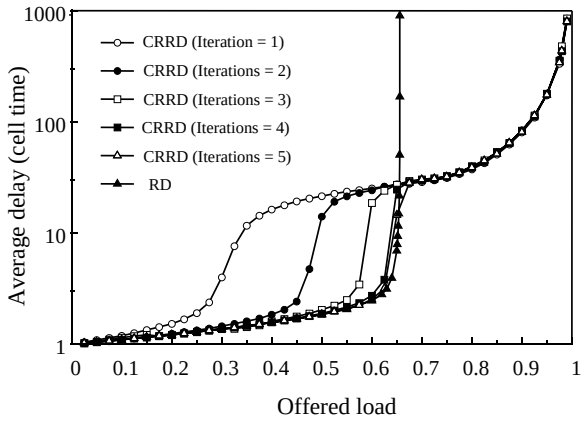


Figure 8: Delay performance of CRRD and RD schemes ($n = m = k = 8$)

store backlogged cells. Under uniform traffic, every VOQ keeps backlogged cells until the idle state is eliminated, i.e., until the stable state is reached. The stable state is defined in [12]. In the stable state, every VOQ is occupied with backlogged cells. In this situation, as illustrated in Figure 5, the desynchronization effect is always obtained. Therefore, even when the offered load is 1.0, no contention occurs in the stable state.

As the number of iterations increases, the delay performance improves when the offered load is less than 0.7, as shown in Figure 8. This is because the matching between VOQs and output links $L_I(i, r)$ within 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 IM in CRRD is not completely achieved, while the complete matching within IM in RD is always achieved, as described in Section 3.1. When the offered load is larger than 0.7, the delay performance of CRRD is not improved by the number of iterations in IM. The number of iterations in IM only improves the matching within IM, but does not improve the matching between IM and CM. The delay performance is improved by the number of iterations in IM when the offered load is not heavy. Figure 8 shows that when the number of iterations in IM increases to 4, the delay performances almost converge.

Figure 9 shows that 99.9% delay of CRRD has the same tendency as the average delay, as shown in Figure 8.

Our simulation shows, as presented in Figure 10, that even when the input traffic is bursty, CRRD provides 100% throughput for any number of iterations. However, the delay performance under the bursty traffic becomes worse than that of the non-bursty traffic at the heavy load condition. The reason for the 100% throughput even for bursty traffic can be explained as described in the Bernoulli traffic discussion above. We assume that the burst length is exponentially distributed as the bursty traffic. In this evaluation, the burst length is set to 10.

By looking at Figure 10, we can make the following observations.

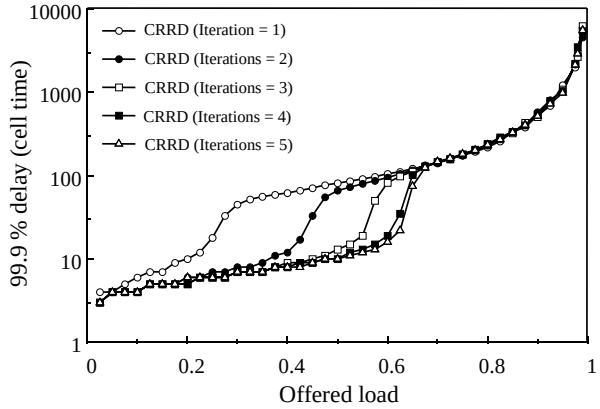


Figure 9: 99.9% delay of CRRD ($n = m = k = 8$)

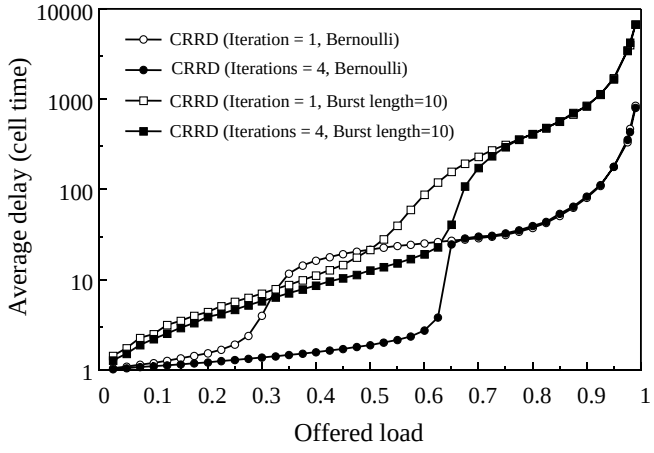


Figure 10: Delay performance of CRRD affected by bursty traffic ($n = m = k = 8$)

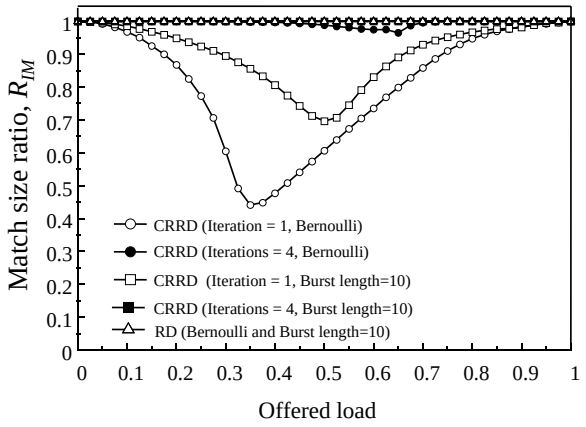


Figure 11: Average match size ratio, R_{IM}, in phase 1 ($n = m = k = 8$)

When the input offered load ρ, where $\rho \leq 0.2$, is very low, the number of iterations does not affect the delay performance significantly. This is because average match size ratios R_{IM} for the matching within IM in phase 1 and R_{IM_CM} for the matching between IM and CM in phase 2 are close to 1.0, as shown in Figures 11 and 12.

The average match size ratios, R_{IM} and R_{IM_CM}, are defined as follows.

R_{IM} is defined as,

$$R_{IM} = \frac{1}{k} \sum_i \frac{M_{IM}(i)}{\min(d_{nev}(i), m)}, \tag{4}$$

where $M_{IM}(i)$ is a match size within $IM(i)$ in phase 1 and $d_{nev}(i)$ is the number of non-empty VOQs in $IM(i)$ at each time slot. When $\min(d_{nev}(i), m) = 0$, $\frac{M_{IM}(i)}{\min(d_{nev}(i), m)}$ is set to 1.0. Note that RD always provides $M_{IM}(i) = \min(d_{nev(i)}, m)$, as is described in Section 3.1.

R_{IM_CM} is defined as,

$$R_{IM\_CM} = \frac{1}{m} \sum_r \frac{\sum_i M_{IM(i)\_CM(r)}}{\sum_i \theta(i, r)}, \tag{5}$$

$$\theta(i, r) = \begin{cases} 1 & \text{if there is a request from } IM(i) \text{ to } CM(r) \\ 0 & \text{otherwise,} \end{cases} \tag{6}$$

and

$$M_{IM(i)\_CM(r)} = \begin{cases} 1 & \text{if there is a grant from } CM(r) \text{ to } IM(i) \text{ in phase 2} \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

When $\sum_i \theta(i, r) = 0$, $\frac{\sum_i M_{IM(i)\_CM(r)}}{\sum_i \theta(i,r)}$ is set to 1.0.

When ρ increases to about 0.4, R_{IM} of CRRD with one iteration for Bernoulli traffic decreases due to contention in the IM. R_{IM} for the bursty traffic is not more affected by ρ than for Bernoulli

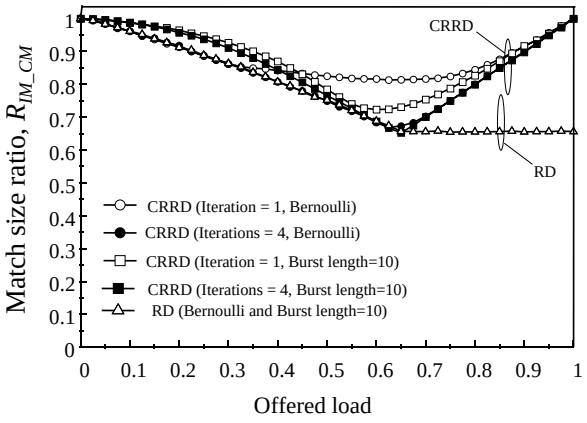


Figure 12: Average match size ratio, R_{IM_CM}, in phase 2 ($n = m = k = 8$)

traffic. The desynchronization effect is more difficult to obtain for Bernoulli traffic than for bursty traffic because the arrival process is more independent. In this load region, where the traffic load is about 0.4, the delay performance with one iteration for Bernoulli traffic is worse than for bursty traffic.

At $0.2 \leq \rho \leq 0.7$, the number of iterations affects the delay performance for Bernoulli traffic, as shown in Figure 10. In this load region, R_{IM} with one iteration for Bernoulli traffic is low, as shown in Figure 11, while R_{IM} with four iterations is nearly 1.0. On the other hand, at $0.4 \leq \rho \leq 0.7$, the number of iterations affects the delay performance for the bursty traffic, as shown in Figure 10. In this load region, R_{IM} with one iteration for the bursty traffic is low, as shown in Figure 11, while R_{IM} with four iterations is nearly 1.0.

When the traffic load becomes very heavy, R_{IM} and R_{IM_CM} with any iterations for both Bernoulli and bursty traffic approach 1.0. In this load region, the desynchronization effect is easy to obtain as described above.

6.1.2 CMSD

This section describes the performance of CMSD under uniform traffic by comparing it to CRRD.

Figure 13 shows that CMSD provides 100% throughput under uniform traffic, as CRRD does, due to the desynchronization effect, as described in Section 5.2. The throughput of CMSD is also independent of the number of iterations in IM.

In Figure 13, the number of iterations in IM improves the delay performance even in heavy load regions in the CMSD scheme. On the other hand, the delay performance of CRRD is improved by the number of iterations in the IM when the offered load is not heavy. As a result, in the heavy load region, the delay performance of CMSD is better than that of CRRD when the number of iterations in IM is large.

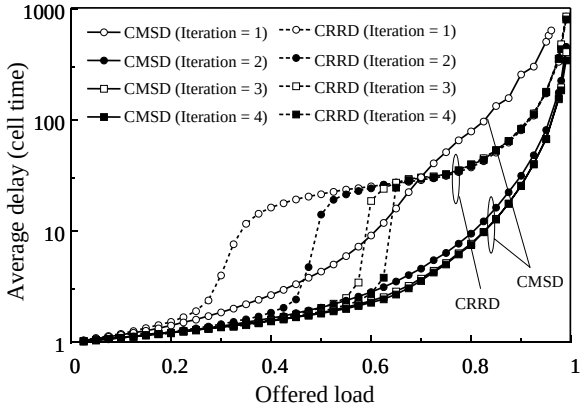


Figure 13: Delay performance of CMSD compared with CRRD ($n = m = k = 8$)

Since the master output-link arbiter of CMSD can choose a VOQ group, not a VOQ itself in CRRD, the master output-link arbiter of CMSD affects the matching between IM and CM more than the output-link arbiter of CRRD. Therefore, CMSD makes the matching between IM and CM more efficient than CRRD when the number of iterations in IM increases. Although this is one of the advantages of CMSD, the main advantage is that CMSD is easier to implement than CRRD when the switch size increases; this will be described in Section 7.

Figure 13 shows that when the number of iterations in IM increases to 4 for CMSD, the delay performance almost converges.

Figure 14 shows via simulation that CMSD also provides 100% throughput under uniform traffic even when a bursty arrival process is considered. The bursty traffic assumption in Figure 14 is the same as that in Figure 10.

In the heavy load region, the delay performance of CMSD is better than that of CRRD under bursty traffic when the number of iterations in IM is large.

6.1.3 Expansion Factor

Figure 15 shows that RD requires an expansion ratio of more than 1.5 to achieve 100% throughput, while CRRD and CMSD need no bandwidth expansion.

6.2 Non-Uniform Traffic

We compared the performance of CRRD, CMSD, and RD using non-uniform traffic. The non-uniform traffic considered here is defined by introducing unbalanced probability, w. Let us consider input port s, output port d, and the offered input load for each input port ρ. The traffic load from input port s

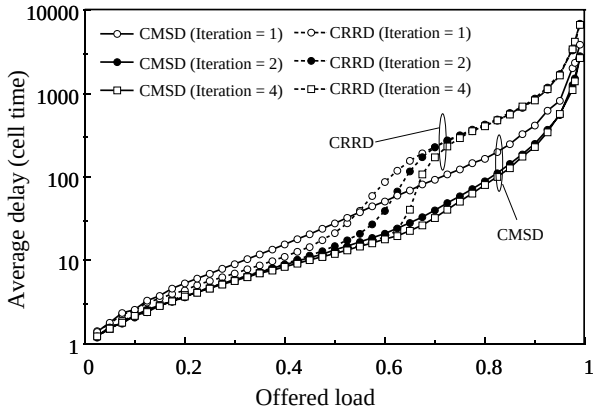


Figure 14: Delay performance of CMSD in bursty traffic compared with CRRD ($n = m = k = 8$)

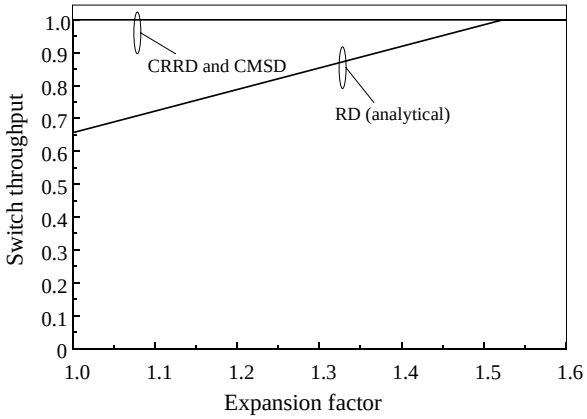


Figure 15: Relationship between switch throughput and expansion factor ($n = k = 8$)

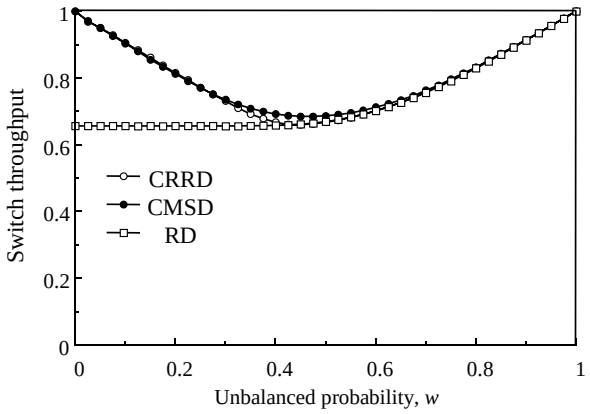


Figure 16: Switch throughput under non-uniform traffic ($n = m = k = 8$)

to output port d, $\rho_{s,d}$ is given by,

$$\rho_{s,d} = \begin{cases} \rho\left(w + \frac{1-w}{N}\right) & \text{if } s = d \\ \rho\frac{1-w}{N} & \text{otherwise.} \end{cases} \tag{8}$$

where $N(= nk)$ is the switch size. Here, the aggregate offered load that goes to output d from all the input ports, ρ_d is given by,

$$\rho_d = \sum_s \rho_{s,d} = \rho\left(w + N \times \frac{1-w}{N}\right) = \rho. \tag{9}$$

When $w = 0$, the offered traffic is uniform. On the other hand, when $w = 1$, it is completely unbalanced. This means that all the traffic of input port s is destined for only output port d, where $s = d$.

Figure 16 shows that the throughput with CRRD and CMSD is higher than that of RD when the traffic is slightly unbalanced.[11] The throughput of CRRD is almost the same as that of CMSD. We assume that enough large numbers of iterations in IM are adopted in this evaluation for both CRRD and CMSD schemes to observe how non-uniform traffic impacts the matching between IM and CM. From $w = 0$ to around $w = 0.4$, the throughput of CRRD and CMSD decreases. This is because the complete desynchronization of the pointers is hard to achieve under unbalanced traffic.[12] However,

[11] Although both results shown in Figure 16 are obtained by simulation, we also analytically derived the upper-bound throughput of RD in Appendix C.

[12] For the same reason, when input traffic is very asymmetric, throughput degradation occurs.

when w is larger than 0.4, the throughput of CRRD and CMSD increases, because the contention at the CM decreases. The reason is that, as w increases, more traffic from $IM(i)$ is destined for $OM(i)$ by using $L_C(r,i)$, and less traffic from $IM(j)$ is destined for $OM(i)$ by using $L_C(r,i)$, where $i \neq j$, according to Eq. (8). That is why contention of $L_C(r,i)$ at $CM(r)$ decreases.

Because of the desynchronization effect, both CRRD and CMSD provide better performance than RD when w is smaller than 0.4. In addition, the throughput of CRRD and CMSD is not worse than that of RD at any w larger than 0.4, as shown in Figure 16.

Developing a new practical dispatching scheme that avoids the throughput degradation even under non-uniform traffic without expanding the internal bandwidth is for further study.

7 Implementation of CRRD and CMSD

This section discusses the implementation of CRRD and CMSD. First, we briefly compare them to RD. Since the CRRD and CMSD schemes are based on round-robin arbitration, their implementations are much simpler than that of RD which needs random generators in the IMs. These generator are difficult and expensive to implement at high speeds [14]. Therefore, the implementation cost for CRRD and CMSD is reduced compared to RD.

The following subsections describe the dispatching scheduling time, hardware complexity, and interconnection-wire complexity for CRRD and CMSD.

7.1 Dispatching Scheduling Time

7.1.1 CRRD

In phase 1, at each iteration, two round-robin arbiters are used. One is an output-link arbiter that chooses one VOQ request out of, at most, nk requests. The other is a VOQ arbiter that chooses one grant from, at most, m output-link grants. We assume that priority encoders are used for the implementation of the round-robin arbiters [14]. Since $nk > m$, the dispatching scheduling time complexity of each iteration in phase 1 is $O(\log nk)$ (see Appendix D) [8]. As is the case of iSLIP, ideally, m iterations in phase 1 are preferable, because there are m output links that should be matched with VOQs in the IM. Therefore, the time complexity in phase 1 is $O(m \log nk)$. However, practically we do not need m iterations. As described in Section 6, simulation suggests that one iteration is sufficient to achieve 100% throughput under uniform traffic. In this case, increasing the number of iterations improves the delay performance.

In phase 2, one IM request is chosen from, at most, k requests. The time complexity at phase 2 is $O(\log k)$.

As a result, the time complexity of CRRD is $O(m \log nk) = O(m \log N)$, where N is the number of switch ports. If we set the number of iterations in phase 1 as i, where $1 \leq i \leq m$, for practical purposes, the time complexity of CRRD is expressed as $O(i \log N)$.

Next, we consider the required dispatching scheduling time of CRRD. The required time of CRRD,

$T_{dis}(CRRD)$, is approximately,

$$T_{dis}(CRRD) = \alpha\{i(\log N + \log m) + \log k\} + \beta_{CRRD}, \tag{10}$$

where α is a constant coefficient, which is determined by device technologies. β_{CRRD} is the transmission delay between arbiters.

7.1.2 CMSD

In phase 1, at each iteration, three round-robin arbiters are used. The first one is a master output-link arbiter that chooses one out of k VOQ group requests. The second one is a slave output-link arbiter that chooses one out of n requests. The third one is a VOQ arbiter that chooses one out of m output-link grants. The slave output-link arbiter can perform its arbitration without waiting for the result in the master output-link arbitration as described in Section 5.1. In other words, both master output-link and slave output-link arbiters operate simultaneously. Since $n \leq m$ is satisfied, the longer arbitration time of either the master output-link arbitration or the VOQ arbitration time is dominant. Therefore, the time complexity of each iteration in phase 1 is $O(\max(\log m, \log k))$. The time complexity in phase 1 is $O(m \max(\log m, \log k))$ if m iterations are adopted.

In phase 2, one is chosen out of k IM requests. The dispatching scheduling time complexity at phase 2 is $O(\log k)$.

As a result, the time complexity of CMSD is $O(m \max(\log m, \log k))$. As in the case of CRRD, for practical purposes, the time complexity of CMSD is expressed as $O(i \max(\log m, \log k))$. When we assume $n = k = m$, the time complexity of CMSD is $O(i \log \sqrt{N})$, which is equivalent to $O(i \log N)$. This is the same order of time complexity as CRRD.

We consider the required dispatching scheduling time of CMSD. With the same assumption as CRRD, the required time of CMSD, $T_{dis}(CMSD)$, is approximately,

$$T_{dis}(CMSD) = \alpha\{i(\max(\log k, \log n) + \log m) + \log k\} + \beta_{CMSD}, \tag{11}$$

where β_{CMSD} is the transmission delay between arbiters.

7.1.3 Comparison of Required Dispatching Scheduling Times

Although the order of the scheduling time complexity of CRRD is the same as that of CMSD, the required times are different. To compare both required times, we define the ratio of $T_{dis}(CMSD)$ to $T_{dis}(CRRD)$ as follows:

$$R_T = \frac{T_{dis}(CMSD)}{T_{dis}(CRRD)}. \tag{12}$$

First, we assume that the transmission delay between arbiters, β_{CRRD} and β_{CMSD}, is negligible compared to the required round-robin execution time to clarify the dependency on n, k, and m so that we can further eliminate the device dependent factor α. Using Eqs. (10), (11), and (12), we obtain the following equation.

$$R_T = \frac{i(\max(\log k, \log m) + \log m) + \log k}{i(\log N + \log m) + \log k}. \tag{13}$$

Table 1 shows the relationship between R_T and i when we assume $n = k = m$. Note that when $n = k = m$, R_T depends only on i, and is independent of N.[13]

Table 1: Ratio of $T_{dis}(CMSD)$ to $T_{dis}(CRRD)$, R_T $(n = k = m)$

i	1	2	3	4	$\cdots$	∞
R_T	0.750	0.714	0.700	0.692	$\cdots$	0.667

When $i = 4$, for example, the required time of CMSD is reduced more than 30% compared to CRRD.

In Eq. (13), we ignored the factors of β_{CRRD} and β_{CMSD}. When β_{CRRD} and β_{CMSD} are large, the reduction effect of CMSD in Table 1 decreases. As the distance between two arbiters gets larger, these factors become significant.

7.2 Hardware Complexity for Dispatching in IM

Since the CM arbitration algorithm of CMSD is the same as that of CRRD, we discuss hardware complexities in IM only.

7.2.1 CRRD

According to implementation results presented in [14], the hardware complexity for each round-robin arbiter is $O(n_{req})$, where n_{req} is the number of requests to be selected by the arbiter.

Each IM has m output-link arbiters and N VOQ arbiters. Each output-link arbiter chooses one out of N requests. The hardware complexity for all the output-link arbiters is $O(mN)$. Each VOQ arbiter chooses one out of m requests. The hardware complexity for all the VOQ arbiters is $O(mN)$. Therefore, the hardware complexity of CRRD in IM is $O(mN)$.

7.2.2 CMSD

Each IM has m master output-link arbiters and mk slave output-link arbiters, N VOQ arbiters. Each master output-link arbiter chooses one out of k requests. The hardware complexity for all the master output-link arbiters is $O(km)$. Each slave output-link arbiter chooses one out of n requests. The hardware complexity for all the slave output-link arbiters is $O(mN)$. Each VOQ chooses one out of m requests. The hardware complexity for all the VOQ arbiters is $O(mN)$. Therefore, the hardware complexity of CMSD in IM is $O(mN)$. Thus, the order of the hardware complexity of CMSD is the same as that of CRRD.

[13] $R_T = (2i + 1)/(3i + 1)$

7.3 Interconnection-Wire Complexity Between Arbiters

7.3.1 CRRD

In CRRD, each VOQ arbiter is connected to all output-link arbiters with three groups of wires. The first group is used by a VOQ arbiter to send requests to the output-link arbiters. The second one is used by a VOQ arbiter to receive grants from the output-link arbiters. The third one is used by a VOQ arbiter to send grants to the output-link arbiters.

As the switch size is larger, the number of wires that connect all the VOQ arbiters with all the output-link arbiters becomes large. In this situation, a large number of crosspoints is produced. That causes high layout complexity. When we implement the dispatching scheduler in one chip, the layout complexity affects the interconnection between transistors. However, if we cannot implement it in one chip due to a gate limitation, we need to use multiple chips. In this case, the layout complexity influences not only the interconnection within a chip, but also the interconnection between chips on PCBs (Printed Circuit Boards). In addition, the number of pins in the scheduler chips of CRRD increases.

Consider that there are X source nodes and Y destination nodes. We assume that each source node is connected to all the destination nodes with wires without detouring as shown in Figure 17. The number of crosspoints for the interconnection wires, N_{xp} between nodes is given by,

$$N_{xp} = \frac{1}{4}XY(X-1)(Y-1). \tag{14}$$

The derivation of Eq. (14) is described in Appendix E. Let the number of crosspoints for interconnection wires between nk VOQs and m output-link arbiters be $N_{xp}(CRRD)$. $N_{xp}(CRRD)$ is determined by,

$$N_{xp}(CRRD) = \frac{3}{4}nkm(nk-1)(m-1). \tag{15}$$

Note that the factor of 3 in Eq. (15) expresses the number of groups of wires as described at the beginning of Section 7.3.1.

7.3.2 CMSD

In CMSD, each VOQ arbiter is connected to its own slave output-link arbiters with three groups of wires. The first group is used by a VOQ arbiter to send requests to the slave output-link arbiters. The second one is used by a VOQ arbiter to receive grants from the slave output-link arbiters. The third one is used by a VOQ arbiter to send grants to the slave output-link arbiters.

In the same way, each VOQ group is connected to all the master output-link arbiters with three groups of wires. The first is used by a VOQ group to send requests to the master output-link arbiters. The second is used by a VOQ group to receive grants from the master output-link arbiters. The third is used by a VOQ group to send grants to the master output-link arbiters.

Let the number of crosspoints for interconnection wires between arbiters in CMSD be $N_{xp}(CMSD)$. $N_{xp}(CMSD)$ is determined by,

$$N_{xp}(CMSD) = \frac{3k}{4}nm(n-1)(m-1) + \frac{3}{4}km(k-1)(m-1). \tag{16}$$

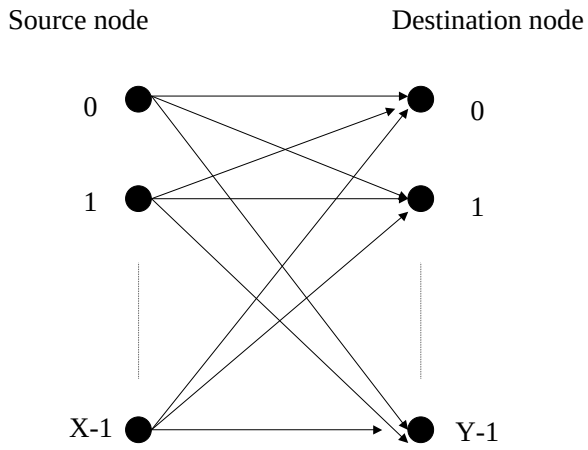


Figure 17: Interconnection-wire model

The first term on the right side is the number of crosspoints of the interconnection wires between n VOQs and m slave output-link arbiters in k VOQ groups. The second term is the number of crosspoints of the interconnection wires between k VOQ groups and m master output-link arbiters.

7.3.3 Comparison of N_{xp}

As the switch size, $N(=nk)$, increases, $N_{xp}(CRRD)$ is much larger than $N_{xp}(CMSD)$. In general, when $n = k = m$, $\frac{N_{xp}(CMSD)}{N_{xp}(CRRD)} = \frac{1}{\sqrt{N}}$. For example, when N=64, 256, and 1024, the number of crosspoints of the interconnection wires of CMSD is reduced by 87.5%, 93.8%, and 96.9%, respectively, compared with that of CRRD. Thus, CMSD can dramatically reduce the number of wire crosspoints.

8 Conclusions

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

First, we have introduced a novel dispatching scheme called CRRD for the Clos-network switch. CRRD provides high switch throughput without increasing internal bandwidth, while only simple round-robin arbiters are employed. Our simulation showed that CRRD achieves 100% throughput under uniform traffic. Even though the offered load reaches 1.0, the pointers of round-robin arbiters that we use at input modules and central modules are completely desynchronized and contention is

avoided.

Second, we have presented CMSD with hierarchical round-robin arbitration, which was developed as an improved version of CRRD to make it more scalable in terms of dispatching scheduling time and interconnection complexity in the dispatching scheduler when the size of the switch increases. We showed that CMSD preserves the advantages of CRRD and achieves more than 30% reduction of the dispatching scheduling time when arbitration time is significant. Furthermore, with CMSD, the number of interconnection crosspoints is dramatically reduced with the ratio of $\frac{1}{\sqrt{N}}$. When $N = 1024$, a 96.9% reduction effect is obtained. This makes CMSD easier to implement than CRRD when the switch size becomes large.

The current work assumes that the dispatching scheduling must be completed within one time slot. However, the constraint is a bottleneck when port speed becomes high. In future work, pipeline-based approaches should be considered. A pipeline-based scheduling approach for crossbar switches is described in [17], which relaxes the scheduling timing constraint without throughput degradation.

References

[1] T. Anderson, S. Owicki, J. Saxe, and C. Thacker, "High speed switch scheduling for local area networks," ACM Trans. Comput. Syst., vol. 11, no. 4, pp. 319-352, Nov. 1993.

[2] 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.

[3] 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.

[4] H. J. Chao and J-S Park, "Centralized Contention Resolution Schemes for a Large-Capacity Optical ATM Switch," Proc. IEEE ATM Workshop'97, pp. 11-16, Fairfax, VA, May 1998.

[5] H. J. Chao , "Saturn: A Terabit Packet Switch Using Dual Round-Robin," Proc. IEEE Globecom 2000, pp. 487-495, Dec. 2000.

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

[8] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, "Introduction to Algorithms," 19th printing, 1997, MIT Press, Section 13.2, page 246.

[9] C. Y. Lee and A. Y. Qruc, "A Fast Parallel Algorithm for Routing Unicast Assignments in Benes Networks," IEEE Trans. Parallel Distributed Sys., vol. 6, no. 3, pp. 329-333, Mar. 1995.

[10] T. T. Lee and S-Y Liew, "Parallel Routing Algorithm in Benes-Clos Networks," Proc. IEEE INFOCOM'96, pp. 279-286, 1996.

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

[12] N. McKeown, V. Anantharam, and J. Walrand, "Achieving 100% Throughput in an Input-Queued Switch," Proc. IEEE INFOCOM'96, pp. 296-302, 1996.

[13] 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.

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

[15] A. Mekkittikul and N. McKeown, "A Practical Scheduling Algorithm to Achieve 100% Throughput in Input-Queued Switches," Proc. IEEE INFOCOM'98, pp. 792-799, 1998.

[16] 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.

[17] E. Oki, R. Rojas-Cessa, and H. J. Chao, "A Pipeline-Based Approach for Maximal-Sized Matching Scheduling in Input-Buffered Switches," IEEE Commun. Let., vol. 5, no. 6, pp. 263-265, Jun. 2001.

[18] 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.

Appendix

A Maximum switch throughput with RD scheme under uniform traffic

To derive Eq. (3), we consider the throughput of output link $OP(j,h)$, $t_{OP(j,h)}$. This is the same as the switch throughput that we would like to derive, because the conditions of all the output links are the same. Therefore,

$$T_{max} = \min\{t_{OP(j,h)}, 1.0\}. \tag{17}$$

First, we define several notations. $Pr^{req}_{VOQ(i,j,h)}$ is the probability that one output link in $IM(i)$, for example $L_I(i,r)$, is used by a $VOQ(i,j,h)$'s request. Here, $Pr^{req}_{VOQ(i,j,h)}$ does not depend on which output link in $IM(i)$ is used for the request, because it is selected randomly. $Pr^{win}_{L_I(i,r)}$ is the probability that the request by $L_I(i,r)$ wins the contention of output link $L_C(r,j)$ at $CM(r)$.

By using $Pr^{req}_{VOQ(i,j,h)}$ and $Pr^{win}_{L_I(i,r)}$, $t_{OP(j,h)}$ is expressed in the following equation.

$$\begin{aligned} t_{OP(j,h)} &= \textstyle\sum_r \sum_i Pr^{req}_{VOQ(i,j,h)} \times Pr^{win}_{L_I(i,r)} \\ &= Pr^{req}_{VOQ(i,j,h)} \times Pr^{win}_{L_I(i,r)} \times m \times k \end{aligned} \tag{18}$$

Here, in the last form of Eq. (18), we consider that $Pr^{req}_{VOQ(i,j,h)}$ and $Pr^{win}_{L_I(i,r)}$ do not depend on both i and r.

Since $IM(i)$ has nk VOQs, each VOQ is equally selected by each output link in $IM(i)$. $Pr^{req}_{VOQ(i,j,h)}$ is given by,

$$Pr^{req}_{VOQ(i,j,h)} = \frac{1}{nk}. \tag{19}$$

Next, we consider $Pr^{win}_{L_I(i,r)}$. $Pr^{win}_{L_I(i,r)}$ is obtained as follows.

$$\begin{aligned} Pr^{win}_{L_I(i,r)} &= \left(\tfrac{1}{k}\right)^{k-1}\tfrac{1}{k} \\ &+ \begin{pmatrix} k-1 \\ k-2 \end{pmatrix}\left(\tfrac{1}{k}\right)^{k-2}\left(1-\tfrac{1}{k}\right)\tfrac{1}{k-1} \\ &+ \begin{pmatrix} k-1 \\ k-3 \end{pmatrix}\left(\tfrac{1}{k}\right)^{k-3}\left(1-\tfrac{1}{k}\right)^{2}\tfrac{1}{k-2} \\ &+ \cdots \\ &+ \begin{pmatrix} k-1 \\ k-i'-1 \end{pmatrix}\left(\tfrac{1}{k}\right)^{k-i'-1}\left(1-\tfrac{1}{k}\right)^{i'}\tfrac{1}{k-i'} \\ &+ \cdots \\ &+ \begin{pmatrix} k-1 \\ 0 \end{pmatrix}\left(1-\tfrac{1}{k}\right)^{k-1}\tfrac{1}{1} \end{aligned} \tag{20}$$

The first term of the right side at Eq. (20) is the winning probability of the request by $L_I(i,r)$ at $CM(r)$ when there are totally k requests for $L_C(r,j)$. The second term is the winning probability of the request by $L_I(i,r)$ at $CM(r)$ when there are $k-1$ requests at $L_C(r,j)$. In the same way, the $(i'+1)$th term is the winning probability of the request by $L_I(i,r)$ at $CM(r)$ when there are $k-i'$ requests at $L_C(r,j)$.

Then, $Pr^{win}_{L_I(i,r)}$ can be expressed in the following equation.

$$Pr^{win}_{L_I(i,r)} = \sum_{i'=0}^{k-1} \begin{pmatrix} k-1 \\ k-i'-1 \end{pmatrix}\left(\frac{1}{k}\right)^{k-i'-1}\left(1-\frac{1}{k}\right)^{i'}\frac{1}{k-i'} \tag{21}$$

By using Eqs. (17), (18), (19), and (21), we can derive Eq. (3).

B Derivation of $1-1/e$ for $k \to \infty$

We sketch the term $\sum_{i=0}^{k-1}$ in Eq. (3) according to the decreasing order in the following:

$$\sum_{i=0}^{k-1} \begin{pmatrix} k-1 \\ k-i-1 \end{pmatrix}\left(\frac{1}{k}\right)^{k-i-1}\left(1-\frac{1}{k}\right)^{i}\frac{1}{k-i}$$

$$
\begin{aligned}
&= \left(1-\frac{1}{k}\right)^{k}\left(1-\frac{1}{k}\right)^{-1} \\
&+\frac{k-1}{k}\left(1-\frac{1}{k}\right)^{k}\left(1-\frac{1}{k}\right)^{-2}\frac{1}{2!} \\
&+\frac{(k-1)(k-2)}{k^2}\left(1-\frac{1}{k}\right)^{k}\left(1-\frac{1}{k}\right)^{-3}\frac{1}{3!} \\
&+\frac{(k-1)(k-2)(k-3)}{k^3}\left(1-\frac{1}{k}\right)^{k}\left(1-\frac{1}{k}\right)^{-4}\frac{1}{4!} \\
&+\cdots
\end{aligned} \tag{22}
$$

When $k \to \infty$,

$$\left(1-\frac{1}{k}\right)^{k} \to \frac{1}{e}. \tag{23}$$

Therefore,

$$
\begin{aligned}
\lim_{k\to\infty}\sum_{i=0}^{k-1}\binom{k-1}{k-i-1}\left(\frac{1}{k}\right)^{k-i-1}&\left(1-\frac{1}{k}\right)^{i}\frac{1}{k-i} \\
&= \frac{1}{e}\left(1+\frac{1}{2!}+\frac{1}{3!}+\frac{1}{4!}+\cdots\right) \\
&= \frac{1}{e}\left(e-1\right) \\
&= 1-\frac{1}{e}.
\end{aligned} \tag{24}
$$

In the above deduction, we use the following equation.

$$e = 2+\frac{1}{2!}+\frac{1}{3!}+\frac{1}{4!}+\cdots \tag{25}$$

C Upper bound of maximum switch throughput with RD scheme under non-uniform traffic

The upper bound of the maximum switch throughput with RD scheme under non-uniform traffic, T_{upper_bound} is determined by the following equation.

$$
\begin{aligned}
T_{upper\_bound} = \min\Bigg\{\frac{m}{n}\Bigg\{&\left(w+\frac{1-w}{k}\right)\sum_{i=0}^{k-1}\binom{k-1}{k-i-1}\left(\frac{1-w}{k}\right)^{k-i-1}\left(1-\frac{1-w}{k}\right)^{i}\frac{1}{k-i} \\
&+(k-1)\left(\frac{1-w}{k}\right)\Bigg[\sum_{i=0}^{k-2}\left(w+\frac{1-w}{k}\right)\binom{k-2}{k-i-2}\left(\frac{1-w}{k}\right)^{k-i-2}\left(1-\frac{1-w}{k}\right)^{i}\frac{1}{k-i} \\
&+\sum_{i=0}^{k-2}\left[1-\left(w+\frac{1-w}{k}\right)\right]\binom{k-2}{k-i-2}\left(\frac{1-w}{k}\right)^{k-i-2}\left(1-\frac{1-w}{k}\right)^{i}\frac{1}{k-i-1}\Bigg]\Bigg\}, 1.0\Bigg\}
\end{aligned} \tag{26}
$$

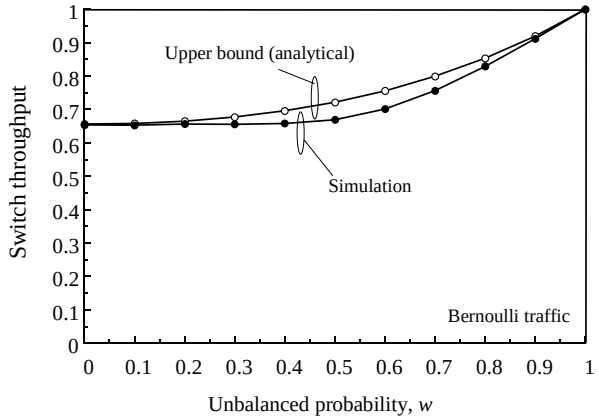


Figure 18: Comparison between simulation and analytical upper-bound results under non-uniform traffic ($n = m = k = 8$)

Eq. (26) can be easily derived in the same way as Eq. (3). However, since the traffic load of each VOQ is different, Eq. (3) has to be modified. The first term of the right side at Eq. (26) is obtained by considering the probability that a heavy-loaded VOQ sends a request to the CM and wins the contention. The second term of the right side is obtained by considering the probability that, when a heavy-loaded VOQ and non-heavy-loaded VOQs send a request to the CM, a non-heavy-loaded VOQ wins the contention. The third term of the right side is obtained by considering the probability that a heavy-loaded VOQ does not send a request to the CM, non-heavy-loaded VOQs send requests to the CM, and a non-heavy-loaded VOQ wins the contention.

Note that Eq. (26) gives the upper-bound of the switch throughput of the RD scheme in the switch model described in Section 2. This is because only one cell can be read from the same VOQ at the same time slot in the switch model described in Section 2. However, we assume that more than one cell can be read from the same VOQ at the same time slot in the derivation of Eq. (26) for simplicity. This different assumption causes different results under non-uniform traffic.[14]

Figure 18 compares the simulation result with the upper-bound result obtained by Eq. (26). As w increases to about 0.6, the difference between the simulation and analytical results increases due to the reason described above. At the region of $0.6 \leq w \leq 1.0$, the difference decreases with w, because the contention probability at CM decreases.

[14] However, under uniform traffic, both results are exactly the same because all the VOQs can be assumed to be in a non-empty state. This does not cause any difference in the results.

D Time complexity of round-robin arbiter

In a round-robin arbiter, the arbitration time complexity is dominated by an N-to-$\log_2 N$ priority encoder, where N is the number of inputs for the arbiter.

The priority encoder is analogous to a maximum or minimum search algorithm. We use a binary search tree, whose computation complexity (and thus the timing) is $O(h)$ where h is the depth of the tree, where h is equal to $\log_2 N$ for a binary tree.

The priority encoder, in the canonical form, has two stages: the AND-gate stage and the OR-gate stage. In the AND-gate stage, there are N AND gates where the largest gate has N inputs. An N-input AND gate can be built by using a number of y-input AND gates where the level of delay is $\log_y N$, i.e., in $\log_2 N$. As for the OR-gate stage, there are $\log_2 N$ OR gates where the largest has $N/2$ inputs. Then the largest AND gate delay is the delay order for the priority encoder.

E Derivation of Eq. (14)

We define an interconnection wire between source node x, where $0 \leq x \leq X-1$ and destination node y, where $0 \leq y \leq Y-1$, as $W(x,y)$. We also define the number of crosspoints that wire $W(x,y)$ has as $n_{xp}(x,y)$.

The total number of crosspoints of the interconnection wires, N_{xp}, between X source nodes and Y destination nodes is expressed by,

$$N_{xp} = \frac{1}{2} \sum_{x=0}^{X-1} \sum_{y=0}^{Y-1} n_{xp}(x,y). \tag{27}$$

Here, the factor of $\frac{1}{2}$ at the right side of Eq. (27) eliminates the double counting of the number of crosspoints.

First consider $W(0,y)$. Since $W(0,0)$ has no crosspoints with other wires, $n_{xp}(0,0) = 0$. Since $W(0,1)$ has $X-1$ crosspoints with $W(1,0)$, $W(2,0)$, ..., $W(X-1,0)$, $n_{xp}(0,1) = X-1$. $W(0,2)$ has $(X-1)\times 2$ crosspoints with $W(1,0)$, $W(2,0)$, ..., $W(X-1,0)$, $W(1,1)$, $W(2,1)$, ..., $W(X-1,1)$. Therefore, $n_{xp}(0,2) = (X-1)\times 2$. In the same way, $n_{xp}(0,y) = (X-1)y$.

Next, consider $W(1,y)$. Since $W(1,0)$ has $Y-1$ crosspoints with $W(0,1)$, $W(0,2)$, ..., $W(0,Y-1)$, $n_{xp}(0,0) = Y-1$. $W(1,1)$ has $X-2$ crosspoints with $W(2,0)$, $W(3,0)$, ..., $W(X-1,0)$, and $Y-2$ crosspoints with $W(0,2)$, $W(0,3)$, ..., $W(0,Y-1)$. Therefore, $n_{xp}(1,1) = (X-2)+(Y-2)$. $W(1,2)$ has $(X-2)\times 2$ crosspoints with $W(2,0)$, $W(3,0)$, ..., $W(X-1,0)$, $W(2,1)$, $W(3,1)$, ..., $W(X-1,1)$, and $Y-3$ crosspoints with $W(0,3)$, $W(0,4)$, ..., $W(0,Y-1)$. Therefore, $n_{xp}(1,2) = (X-2)2+(Y-3)$. In the same way, $n_{xp}(1,y) = (X-2)y + (Y-1-y)$.

Then we can derive the following:
$n_{xp}(2,y) = (X-3)y + (Y-1-y)2$,
$n_{xp}(3,y) = (X-4)y + (Y-1-y)3$,
$n_{xp}(4,y) = (X-5)y + (Y-1-y)4$,
...

In general, $n_{xp}(x, y)$ is given by,

$$n_{xp}(x, y) = (X - 1 - x)y + (Y - 1 - y)x. \tag{28}$$

We substitute Eq. (28) into Eq. (27). The following equation is obtained.

$$\begin{aligned} N_{xp} &= \tfrac{1}{2} \textstyle\sum_{x=0}^{X-1} \sum_{y=0}^{Y-1} \{(X - 1 - x)y + (Y - 1 - y)x\} \\ &= \tfrac{1}{4} XY(X - 1)(Y - 1) \end{aligned} \tag{29}$$