PAPER

# Routing Algorithms for Packet/Circuit Switching in Optical Multi-$\log_2 N$ Networks

**Yusuke FUKUSHIMA**[†a)], ***Student Member***, **Xiaohong JIANG**[†], ***Nonmember***, ***and*** **Susumu HORIGUCHI**[†], ***Member***

**SUMMARY** The multi-$\log_2 N$ network architecture is attractive for constructing optical switches, and the related routing algorithms are critical for the operation and efficiency of such switches. Although several routing algorithms have been proposed for multi-$\log_2 N$ networks, a full performance comparison among them has not been published up to now. Thus, we rectify this omission by providing such a comparison in terms of blocking probability, time complexity, hardware cost and load balancing capability. Notice that the load balance is important for reducing the peak power requirement of a switch, so we also propose in this paper a new routing algorithm for optical multi-$\log_2 N$ networks to achieve a better load balance.

***key words:*** *optical switch networks, multi-$\log_2 N$ networks, $\log_2 N$ networks, directional coupler, blocking network*

## 1. Introduction

The optical switching networks (or switches), that can switch optical signals in optical domain at ultra-high speed, will be the key supporting elements for the operation and efficiency of future high-capacity optical networks and high-speed interconnected systems. The multi-$\log_2 N$ network architecture, which is based on the vertical stacking of multiple $\log_2 N$ networks [1], is attractive for constructing the optical switches due to its small depth, absolute loss uniformity, etc.

Since a multi-$\log_2 N$ network consists of multiple copies (planes) of a $\log_2 N$ network, so for each request (e.g., a connection request between an input-output pair) we have to select a plane to route this request based on a specified strategy. We call such a plane selection strategy as the routing algorithm for multi-$\log_2 N$ networks. The routing algorithm is important for the operation and efficiency of a switch, since it directly affects the overall switch hardware cost and also the switching speed. By now, several routing algorithms have been proposed for multi-$\log_2 N$ networks, such as random routing [2]–[4], packing [2], save the unused [2], etc. It is notable that the available results on routing algorithms of multi-$\log_2 N$ networks mainly focus on the nonblocking condition analysis when a specified routing algorithm is applied [1], [2], [5]–[8]. The recent results in [2] indicate that although some routing algorithms are apparently very different, such as save the unused, packing, minimum index, etc., they actually require a same number of planes to guarantee the nonblocking property of multi-$\log_2 N$ networks. The results in [2] also imply that a very high hardware cost (in terms of number of planes) is required to guarantee the nonblocking property, which makes the nonblocking design of multi-$\log_2 N$ networks impractical for real applications. The blocking design of multi-$\log_2 N$ networks is a promising approach to significantly reducing the hardware cost [4]. However, little literature is available on the performance of the available routing algorithms when they are adopted in the blocking network design [3], [4]. In particular, no work is available on the detailed performance comparison among the available routing algorithms when they are applied to a blocking multi-$\log_2 N$ network (e.g., a multi-$\log_2 N$ network with a less number of planes required by its nonblocking condition).

It is notable that the load-balancing capability of a routing algorithm is also important for the multi-$\log_2 N$ networks, since it directly affects the peak power requirement and power dissipation requirement (mainly determined by the maximum number of connections simultaneously supported by a plane). Kabacinski et al. [9] mentioned that heavy loading on a plane results in earlier failures of switches when switches are built using usage-sensitive technology. However, little available algorithms take into account the load-balancing issue in the routing process.

In this paper, to address the above two main issues, we propose a routing algorithm with good load-balancing capability and also fully compare all routing algorithms in terms of blocking probability, hardware cost, complexity and load-balancing capability for both optical packet switching and optical circuit switching technologies. The rest of this paper is organized as follows. Section 2 illustrates the structure and the features of multi-$\log_2 N$ networks. Section 3 introduces the available routing algorithms and our routing algorithm. Section 4 and Sect. 5 provide the comparison among the routing algorithms for optical packet switching and optical circuit switching, respectively, and finally, the Sect. 6 concludes this paper.

## 2. Multi-$\log_2 N$ Networks

### 2.1 Architecture and Features

The multi-$\log_2 N$ network architecture was first proposed by Lea [1]. A multi-$\log_2 N$ network consists of multiple vertically stacked planes as shown in Fig. 1, where each plane is just a banyan class ($\log_2 N$) network [1]–[8] illustrated

Manuscript received April 30, 2008.
Manuscript revised July 23, 2008.

[†]The authors are with the Graduate School of Information Sciences, Tohoku University, Sendai-shi, 980-8579 Japan.
a) E-mail: yusuke@ecei.tohoku.ac.jp

DOI: 10.1093/ietcom/e91–b.12.3913

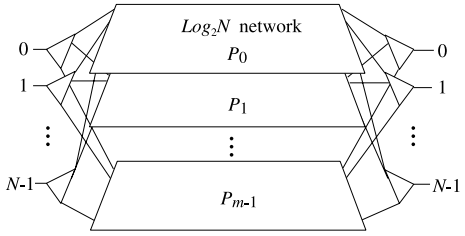



**Fig. 1** A multi-$\log_2 N$ network with $m$-planes. Each request will be routed through one of $m$ planes, $p_0$, $p_1$, …, $p_{m-1}$.

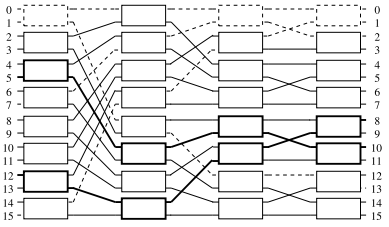


**Fig. 2** A 16×16 banyan network (even number of stages) with a crosstalk scenario.

in Fig. 2. The multi-$\log_2 N$ networks have several attractive properties, such as good fault-tolerance capability, a small network depth and the absolute loss uniformity, which make them attractive for building the *directional coupler* (DC)-based optical switches to provide nano-second order switch speed [1], [4], [6], [14].

Although the DC technology can support much higher switching speed than others (e.g. MEMS) and can switch multiple wavelengths simultaneously, it may suffer from the crosstalk problem when two optical signals pass through a common DC at the same time. A simple and cost effective approach to guaranteeing a low crosstalk in DC-based optical multi-$\log_2 N$ networks is to apply the node-disjoint (or DC-disjoint) constraint to all the connections in this network [3], [4], [8]. Thus, this paper focuses on the optical multi-$\log_2 N$ networks with the node-disjoint constraint. It is notable that the above node-disjoint constraint will cause the node-blocking problem, which happens when two optical signals go through a common node (DC) at the same time and one of them will be blocked. Since link-blocking between two signals will definitely cause node-blocking between them, but the reverse may not be true. Thus, we only need to consider the node-blocking issue in the routing process of optical multi-$\log_2 N$ networks.

When input ports have heavy traffic, the plane selection based on load-balancing is effective in reducing the peak power demand, which directly affects the cost of power equipment. The number of requests on a plane is critical for the switch device built by usage-sensitive technology [9], since it may lead to a large number of disconnections when a heavy loaded plane is broken. We refer to such the number of requests allocated to a plane as the *plane load* hereafter.

### 2.2 Notations

For the convenience of explanations, we here define some notations. We use the notation $\mathrm{Log}_2(N, m)$ to denote a multi-$\log_2 N$ network with $N$ input (output) ports and $m$ planes, and we number its planes from the top to the bottom as $p_0, p_1 \ldots, p_{m-1}$ as shown in Fig. 1. We define a request as an one-to-one (unicast) connection request for a $\mathrm{Log}_2(N, m)$ and denote a request between input $x$ and output $y$ as $\langle x, y\rangle$. We further define a *request frame* as the set of all requests to the network and denote it as

$$\begin{pmatrix} x_0 & x_1 & \cdots & x_{k-1} \\ y_0 & y_1 & \cdots & y_{k-1} \end{pmatrix}, \tag{1}$$

where $0 < k \leq N$ ($k$ is number of requests). An example of request frame is given as follows.

$$\begin{pmatrix} 0 & 1 & 5 & 7 & 12 & 15 \\ 1 & 13 & 10 & 2 & 8 & 0 \end{pmatrix} \tag{2}$$

For the requests of the request frame in Eq. (2), the node-blocking (blocking for short) scenario among them is illustrated in Fig. 2, where the blocking happens between $\langle 0, 1\rangle$ and $\langle 1, 13\rangle$, between $\langle 0, 1\rangle$ and $\langle 7, 2\rangle$ and between $\langle 0, 1\rangle$ and $\langle 15, 0\rangle$. In this scenario, $\langle 0, 1\rangle$ will be blocked by the other three requests.

Notice that a plane of the multi-$\log_2 N$ networks offers only one unique path to each connection, so the blocking will happen between two connections in the plane if their paths share a common node and the blocked one must be set up through another plane. It is also notable that although packing routing paths can save empty planes for other requests (e.g. $\langle 7, 2\rangle$ and $\langle 15, 0\rangle$ can be set up into a plane), it lead to heavy loading on a plane. For a set of requests to be setup, how to choose a plane for each request under the node-disjoint constraint requires a routing algorithm, as explained in the next section.

In this paper, we consider both of *uniform* and *non-uniform* traffic models. Let $\mathcal{A}$ denote the set of output ports. In the uniform traffic model, $y_k$ is randomly selected from $\mathcal{A}$. In the non-uniform traffic model, we consider well-known hotpot traffic, which implies for a number of simultaneous requests for an specific output port. In the hotspot traffic model, we assume $y_k$ will be $x_k$ with probability 0.5, otherwise $y_k$ is randomly selected from $\mathcal{A}\backslash x_k$.

## 3. Routing Algorithms

In this section, we first introduce the seven available routing algorithms for a multi-$\log_2 N$ network and then propose a new one with a better load-balancing capability. For a given request frame and its set of requests, a routing algorithm will try to route these requests one by one sequentially based on both the node-blocking constraint and a specified strategy. We also include one possible routing result on $\mathrm{Log}_2(16,3)$ with Eq. (2) for each algorithm, and use a notation $\mathcal{R}(p_i)$ to denote the set of requests established in $p_i$. The eight routing

algorithms and their main strategies are summarized in the Table 1.

### 3.1 Traditional Routing Algorithms

**(i)** Random ($R$). For a request, the $R$ algorithm [2], [3] selects a plane randomly among all the available planes for this request, if any. One possible routing result is as follows:
$\mathcal{R}(p_0) = \{\langle 1, 13\rangle, \langle 15, 0\rangle\}$
$\mathcal{R}(p_1) = \{\langle 0, 1\rangle\}$
$\mathcal{R}(p_2) = \{\langle 5, 10\rangle, \langle 7, 2\rangle, \langle 12, 8\rangle\}$

**(ii)** Packing ($P$). Based on the $P$ algorithm [4], [10]–[12], we always try to route a request through the busiest plane first, the second busiest plane second, and so on until the first available one emerges. The $P$ algorithm has been well-known as an effective algorithm to reduce the request blocking probability, but it may introduce a very heavy traffic load (in terms of number of requests) to a plane. One possible routing result is as follows:
$\mathcal{R}(p_0) = \{\langle 0, 1\rangle, \langle 5, 10\rangle, \langle 12, 8\rangle\}$
$\mathcal{R}(p_1) = \{\langle 1, 13\rangle, \langle 7, 2\rangle, \langle 15, 0\rangle\}$
$\mathcal{R}(p_2) = \{\}$

**(iii)** Minimum index ($MI$). The $MI$ algorithm [2] always try to route a connection through the first plane $p_0$ first, second plane $p_1$ second and so on until the the first available plane appears. One possible routing result is as follows:
$\mathcal{R}(p_0) = \{\langle 0, 1\rangle, \langle 5, 10\rangle, \langle 12, 8\rangle\}$
$\mathcal{R}(p_1) = \{\langle 1, 13\rangle, \langle 7, 2\rangle, \langle 15, 0\rangle\}$
$\mathcal{R}(p_2) = \{\}$

**(iv)** Save the unused ($STU$). To route a request based on the $STU$ algorithm [2], we do not select the empty plane(s) unless we can not find an occupied plane that can route the request. In this paper, $STU$ algorithm randomly selects a plane from occupied planes, otherwise $STU$ selects empty plane. One possible routing result is as follows:
$\mathcal{R}(p_0) = \{\langle 0, 1\rangle, \langle 12, 8\rangle\}$
$\mathcal{R}(p_1) = \{\langle 1, 13\rangle, \langle 5, 10\rangle, \langle 7, 2\rangle, \langle 15, 0\rangle\}$
$\mathcal{R}(p_2) = \{\}$

**(v)** Cyclic static ($CS$). The $CS$ algorithm [2] always keep a pointer to last used plane [2]. To route a new request, it checks the pointed plane first, then follows the same manner as that of the $MI$ algorithm. The difference between the $MI$ and $CS$ is that starting point of latter is not fixed and it depends on the last used plane. One possible routing result is as follows:
$\mathcal{R}(p_0) = \{\langle 0, 1\rangle\}$
$\mathcal{R}(p_1) = \{\langle 1, 13\rangle, \langle 5, 10\rangle, \langle 7, 2\rangle, \langle 12, 8\rangle, \langle 15, 0\rangle\}$
$\mathcal{R}(p_2) = \{\}$

**(vi)** Cyclic dynamic ($CD$). The $CD$ algorithm [2] is almost the same as $CS$ algorithm, and the only difference is that the $CD$ algorithm always check the next plane of the pointed one first. One possible routing result is as follows:
$\mathcal{R}(p_0) = \{\langle 0, 1\rangle\}$
$\mathcal{R}(p_1) = \{\langle 1, 13\rangle, \langle 7, 2\rangle, \langle 15, 0\rangle\}$
$\mathcal{R}(p_2) = \{\langle 5, 10\rangle, \langle 12, 8\rangle\}$

**Table 1** Routing algorithms for Log$_2(N, m)$

Available routing algorithms

| Algorithm | Description |
|---|---|
| Random ($R$) | Choose a plane randomly from available planes |
| Packing ($P$) | Choose a busiest, yet available, planes |
| Minimum index ($MI$) | For each request, route in the order $p_0, p_1, \ldots,$ until the first available one emerges |
| Save the unused ($STU$) | Do not route through an empty planes unless there is no choice |
| Cyclic static ($CS$) | If $p_s$ was used last, try copy $p_s, p_{s+1}, \ldots,$ until the first available one emerges |
| Cyclic dynamic ($CD$) | If $p_s$ was used last, try $p_{s+1}, p_{s+2}, \ldots,$ until the first available one emerges |
| Danilewicz's algorithm ($D$) | Choose plane that new connection will block the fewest number of future requests of all planes |

Proposed routing algorithm

| Algorithm | Description |
|---|---|
| Load sharing ($LS$) | Choose a least occupied, yet available, planes |

It is interesting to notice that although the above six routing algorithms apparently different, a recent study in [2] revealed that they actually require a same number of planes to guarantee the nonblocking property of a multi-log$_2N$ network. The results in [2] also imply that the nonblocking design of multi-log$_2N$ networks is not very practical since it requires a very high hardware cost (in terms of number of required planes).

**(vii)** Danilewicz ($D$) algorithm. Danilewicz et al. [6] recently proposed an novel routing algorithm for multi-log$_2N$ networks to guarantee the nonblocking property with a reduced number of planes. The main idea of this algorithm is to select the plane that new connection will block the fewest number of future requests of all planes. It is notable, however, the time complexity of this algorithm is significantly higher than the above six routing algorithms (please refer to section 4.5). One possible routing result is as follows:
$\mathcal{R}(p_0) = \{\langle 0, 1\rangle\}$
$\mathcal{R}(p_1) = \{\langle 1, 13\rangle, \langle 5, 10\rangle, \langle 7, 2\rangle, \langle 12, 8\rangle, \langle 15, 0\rangle\}$
$\mathcal{R}(p_2) = \{\}$

Notice that the blocking design of multi-log$_2N$ networks is a promising approach to dramatically reducing their hardware cost [4] without introducing a significant blocking probability. However, the available study on the above seven algorithms mainly focus on their corresponding nonblocking conditions analysis and little literature is available on the performance of these routing algorithms when they are adopted in the blocking network design [3], [4] (e.g., a multi-log$_2N$ network with a less number of planes required by its nonblocking condition). It is an interesting question that although the nonblocking conditions of the available seven routing algorithms are the same, whether their performance is still the same when that they are applied to a blocking multi-log$_2N$ network. To answer this question, in the next section we will conduct a detailed performance

comparison among the available routing algorithms in terms of blocking probability, hardware cost, complexity and load-balancing capability.

### 3.2 A New Load-Balancing Algorithm

It is notable that the load-balancing capability of a routing algorithm is also important for the multi-$\log_2 N$ network, since it directly affects the peak power requirement and power dissipation requirement (mainly determined by the maximum number of connections simultaneously supported by a plane).

We will see in the next section that although the most available algorithms for multi-$\log_2 N$ networks can achieve a low blocking probability, they usually result in a very uneven load distribution among all planes. To provide a better load balance among all planes of a multi-$\log_2 N$ network, we propose new routing algorithm, namely the *load sharing* algorithm as follows.

**(viii)** Load sharing (*LS*). The main idea of the *LS* algorithm, as contrasted to the *P* algorithm, is to route request in the least occupied plane first, the second least occupied plane and so on until the first available one emerges. One possible routing result is as follows:
$\mathcal{R}(p_0) = \{\langle 0, 1\rangle, \langle 12, 8\rangle\}$
$\mathcal{R}(p_1) = \{\langle 1, 13\rangle, \langle 7, 2\rangle\}$
$\mathcal{R}(p_2) = \{\langle 5, 10\rangle, \langle 15, 0\rangle\}$

## 4. Experimental Results for Packet Switching

Optical packet switching (OPS for short) enables the switching of optical data at the granularities of packets in the optical domain. In such a switch, all the arriving packets in a time slot are synchronized and routed simultaneously by a rapid reconfiguration on a packet-by-packet basis to accommodate dynamic traffic.

To support the OPS in a multi-$\log_2 N$ switch, the switch requires the faster routing algorithm to achieve the rapid reconfiguration for dynamic traffic. The load balancing for each plane is also important and very effective in reducing the cost of power equipment on a plane because synchronizing switched packets on one plane can increase the peak power demand, and each power supply on a plane have to support the maximum power consumption.

In this section, we conduct an extensive simulation study for OPS switch to compare the performance of the above eight routing algorithms in terms of blocking probability, time complexity, hardware cost and load balancing capability.

### 4.1 Simulation Setting and Parameters

Our simulation program consists of two main modules: the request frame generator and the request router. The request frame generator generates request frames based on the probability that an input/output port is busy, say $r$. The order of all requests in a frame is determined randomly. For a request frame, the request router module will apply a specified routing algorithm to route the requests in the frame one by one sequentially according their order. We summarize the simulation procedures and parameters as follows:

Simulation procedure for packet switching

1: Request frame generator generates a request frame $\mathcal{F}$ with $r$ and the specified traffic model
2: Determine the order of requests randomly and set the last request as *tagged path*
3: **for** each routing algorithm **do**
4: Initialize the request router
5: Each request in $\mathcal{F}$ is set up based on the specified routing strategy by the request router
6: **if** *Tagged path* is blocked at least once **then**
7: Count blocking
8: **end if**
9: Record $C_{max}$ and $C_{min}$
10: **end for**
11: **return** simulation results

| Simulation parameters | |
|---|---|
| Network size($N$) | 8, 16, 32, 64, 128, 256 |
| #planes ($m$) | 1,2, . . ., until blocking probability $\leq 10^{-6}$ |
| Work Load ($r$) | 0.5, 0.6, 0.7, 0.8, 0.9, 1.0 |
| Iteration time | $10^7$ |
| $C_{max}$ ($C_{min}$) | Maximum (minimum) plane load |

Our simulation program is implemented in C on a cluster workstation — Opteron 2.0 GHz cluster.

### 4.2 Blocking Probability

The blocking probability of OPS, BP for short, is usually measured as the ratio of overall dropped packets to the total number of packets. This definition actually measures the average blocking probability among all packets. It is notable, however, that in a real OPS switch, the blocking probability for all the packets in a frame is related to their orders of set up and thus not the same, where the last packet to be set up has the highest probability to be blocked (refer to as the worst BP hereafter). This worst BP is very useful for understanding the overall blocking behavior of a routing algorithm. The *tagged-request* based analysis proposed in [3], [4], [7] provides us a good way to measure this worst BP, so we adopt the tagged-request based simulation here. For each request frame, we define the tagged-request as the last request in the frame (e.g., for the frame in the example (2), the tagged request is $\langle 15, 0\rangle$). For reference, we also included in our simulation the upper bound and the lower bound[†] on the blocking probability established in [4]. The blocking probability, BP for short, is calculated as follows.

$$BP = \frac{\text{blocked times of the tagged request}}{\text{iteration times}} \tag{3}$$

[†]The upper/lower bound are the theoretical limits calculated by a given request frame.

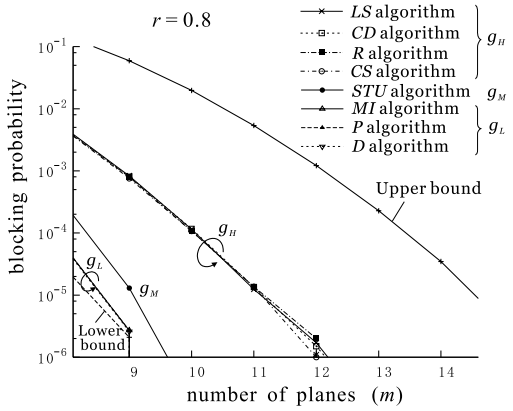



**Fig. 3** Blocking probability on Log$_2$(256,$m$) under the uniform traffic.

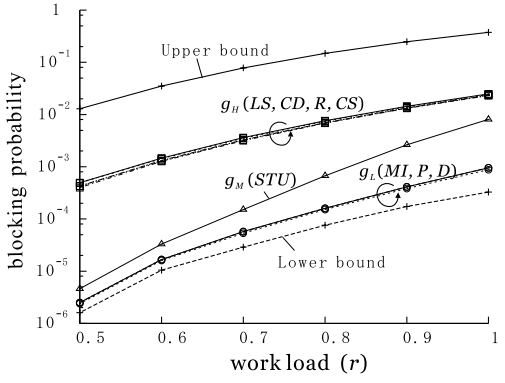



**Fig. 4** Blocking probability versus the work loads on Log$_2$(128,7) under the uniform traffic.

### 4.2.1 Uniform Traffic

In the uniform traffic model, the result shown in Fig. 3 indicates that all routing algorithms could be roughly divided into three groups based on BP. The group of algorithms with high BP, say $g_H$, includes $R$, $LS$, $CS$ and $CD$ algorithms. The group of algorithms with middle BP, say $g_M$, includes $STU$ algorithm. The group of algorithm with low BP, say $g_L$, includes $MI$, $P$ and $D$ algorithms. It is interesting to see from the above figure that although the nonblocking conditions of the traditional seven routing algorithms in Table 1 is the same, their blocking probability are very different.

The main reason of the above BP variation is the difference of the blocking path distribution. Since a connection path is blocked by at least one blocking path on a plane, distribution of blocking paths affects BP. For $g_L$, it is clear that routing strategy is to pack blocking paths as much as possible. In contrast, since the routing strategies of $g_H$ don't care if a request is a blocking path or not, blocking paths can be randomly set up into planes regardless of their different routing strategies. For $g_M$, the routing strategy of $STU$ is the same as $R$ algorithm for $(m - \epsilon)$ planes, where $\epsilon$ is the number of empty planes. It is notable that $g_L$ is very close to the lower bound of BP while $g_H$ has much lower BP than

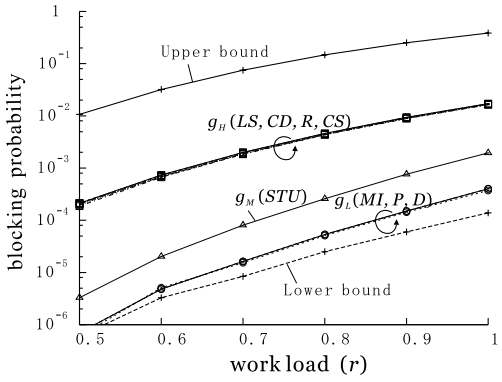



**Fig. 5** Blocking probability versus the work loads on Log$_2$(256,8) under the uniform traffic.

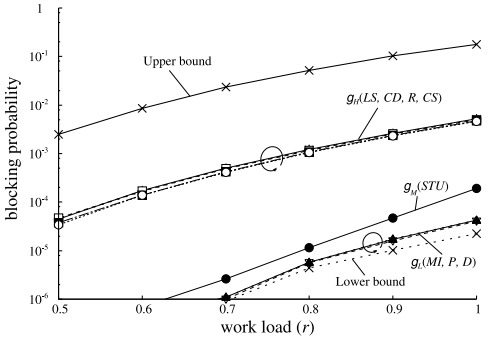



**Fig. 6** Blocking probability versus the work loads on Log$_2$(128,8) under the uniform traffic.

the upper bound.

It is also interesting to notice from Fig. 4 and Fig. 5 that although the BP of all three groups algorithms grows as the work load increases, the BP of $g_M$ is actually very similar to that of the $g_L$ when the work load is low (e.g., when $r < 0.6$), while the BP of $g_M$ is similar to the BP of $g_L$ when the work load is high (e.g., when $r > 0.9$).

To understand the behavior of $g_M$ more clearly, we also include the another configuration Log$_2$(128,8) in Fig. 6. The BP of $STU$ follows $g_L$ as compared to Log$_2$(128,7) shown in Fig. 4. It can be seen that the number of planes significantly affects the saving plane capability of $STU$ algorithm. Therefore, if we want to use $STU$ algorithm to reduce BP, we have to implement enough number of planes.

### 4.2.2 Non-uniform Traffic

For the non-uniform traffic model, the Fig. 7 illustrates the BP for $r = 0.8$ under different number of planes, and Fig. 8 shows the BP for $m = 14$ under various work loads. The non-uniform traffic significantly increases BPs of all algorithms here. We can also divide routing algorithm into three groups, $g_H$, $g_M$ and $g_L$, as Sect. 4.2.1.

It is interesting to notice that the BP of $STU$ algorithm in $g_M$ follows $g_L$ as contrasted with uniform traffic case. To explain the non-uniform traffic, we focus on the frequently generated requests $\langle i, i\rangle$ and $\langle i{+}1, i{+}1\rangle$, where $i$ is even num-

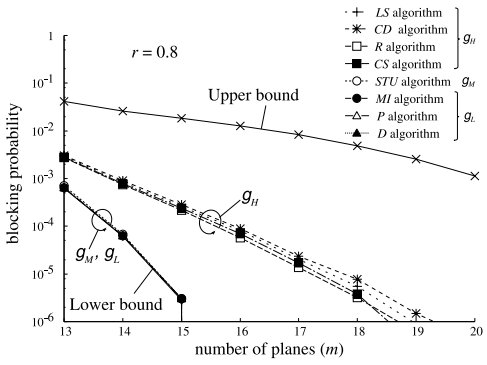



**Fig. 7** Blocking probability on $\mathrm{Log}_2(256,m)$ under the non-uniform traffic.

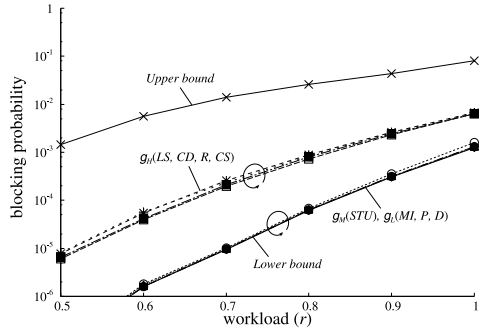



**Fig. 8** Blocking probability versus the work loads on $\mathrm{Log}_2(256,14)$ under the non-uniform traffic.

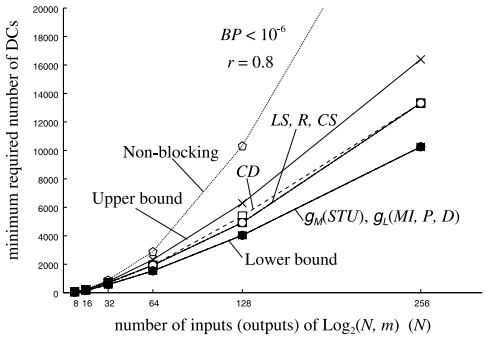



**Fig. 9** Minimum number of planes with blocking probability $< 1.0^{-6}$, $r = 0.8$, under the uniform traffic.

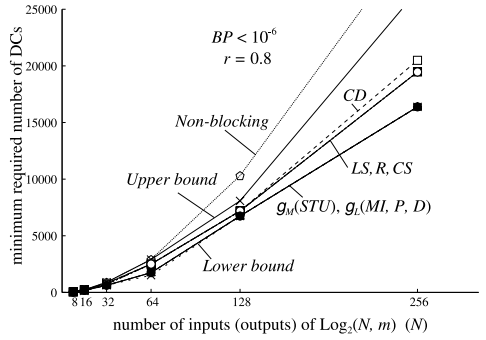



**Fig. 10** Minimum number of planes with blocking probability $< 1.0^{-6}$, $r = 0.8$, under the non-uniform traffic.

ber. In such two requests, since both of requests can be blocked by at least one blocking path simultaneously, therefore it could be more difficult to pack such requests into the same plane. It can be observed that the BP of $g_L$ follows the lower bound. In other words, such requests need to be set up on different planes, and it significantly increases BP by distributing blocking paths. For $g_H$, BP is further increased due to their distribution capability of blocking paths. In contrast, it is notable that BP of $g_M$ is similar to $g_L$ since packing requests became more difficult for the non-uniform traffic pattern.

### 4.3 Hardware Cost (Number of Switch Elements)

In this section, we compare the required number of switch elements by different algorithms. For reference, we also include the required cost for non-blocking condition. When the upper limit on BP is set as $10^{-6}$, the minimum number of directional couplers (DCs) required by different algorithms under both uniform and non-uniform traffic are shown in Fig. 9 and Fig. 10, respectively.

From Fig. 9 and Fig. 10, we can easily notice that the required hardware cost of all routing algorithms is much less than their counterpart in nonblocking condition even when a high requirement on BP is applied under the uniform traffic. We can also observe the hardware cost of routing algorithms in $g_M$ and $g_L$ are all similar and close to the lower bound.

### 4.4 Load Balance and Peak Power Requirement

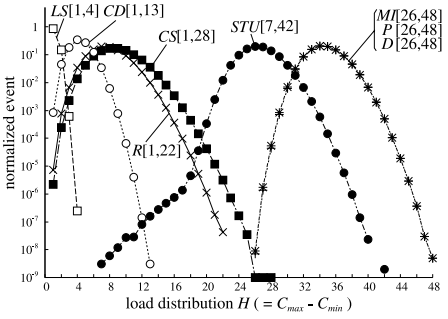
Let $C_{max}$ ($C_{min}$) denotes the maximum (minimum) number of connections in one plane. We refer to the difference between $C_{max}$ and $C_{min}$ as the load distribution index, which is denoted by $H$. The smaller $H$ indicates the better load balancing. To guarantee a reliable evaluation result, we simulated the plane load of OPS with as many as $10^9$ randomly generated request frames, and all the events were fully considered in the final evaluation. To distinguish the difference among algorithms more clearly, we also use the notation $[a,b]$ to denote an interval of occurred events, where $a$ and $b$ are the maximum and the minimum value of all events.

#### 4.4.1 Uniform Traffic

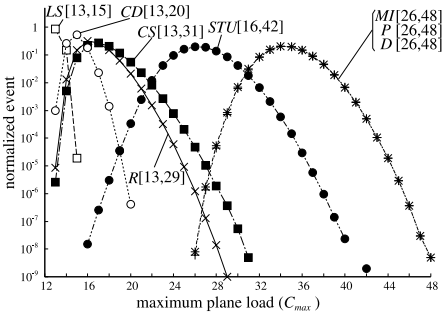
Figure 11(a) and (b) illustrate the distribution of $H$ and $C_{max}$ for each routing algorithm on a $\mathrm{Log}_2(128,10)$ network with $r = 1.0$, respectively.

It can be seen that although the routing algorithms in $g_H$ are very similar in other comparisons, their load balancing capabilities are very different. Especially, load balancing can be further improved by *LS* algorithm, while the algorithms in $g_L$ (e.g, *P* and *D* algorithm) may suffer from very heavy plane load.

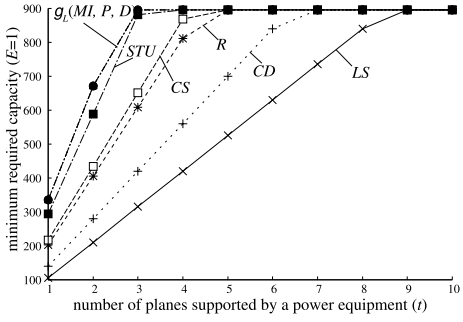
It is notable that the distribution of $C_{max}$ affects the peak power/power dissipation requirement. The result in the




(a) Distribution of the plane load balance index $H$.

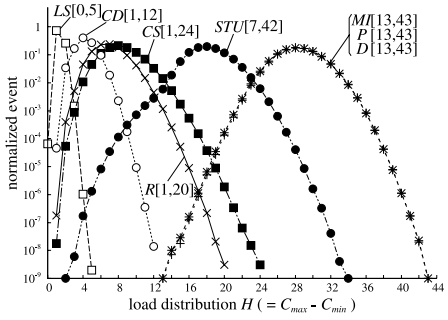



(b) Distribution of the maximum plane load.

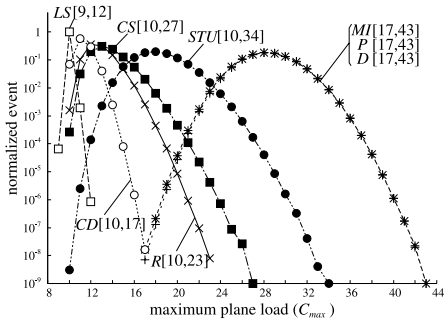



(c) Minimum required capacity of power equipment.

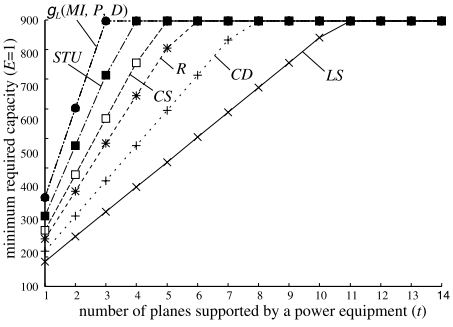
**Fig. 11** Plane load and peak power requirement on Log$_2$(128,10) under the uniform traffic, $r$ = 1.0.




(a) Distribution of the plane load balance index $H$.




(b) Distribution of the maximum plane load.




(c) Minimum required capacity of power equipment.

**Fig. 12** Plane load and peak power requirement on Log$_2$(128,14) under the non-uniform traffic, $r$ = 1.0.

Fig. 11(b) indicates that the $LS$ algorithm can efficiently reduce the peak power requirement, because the $C_{max}$ of $LS$ is the closest the average value of 128/10 = 12.8. In contrast, the algorithms in $g_L$ (e.g, $P$ and $D$ algorithm) have a much higher power requirement. It is also interesting to see that the required number of planes in $g_M$ ($STU$) and in $g_L$ is the same as the lower bound shown in Fig. 9. However, load balance of $STU$ algorithm is better than $g_L$.

We also discuss here how much capacity is needed for each power equipment. Let $\mathcal{P}$ denote the peak power requirement. When a power equipment supports $t$ planes (the total number of power equipments is $\lceil \frac{m}{t} \rceil$), the $\mathcal{P}$ is obtained by the following equation.

$$\mathcal{P} \geq min(tC, rN) \cdot Elog_2N, \tag{4}$$

where $C$ and $E$ denote the maximum $C_{max}$ and the power consumption of a DC, respectively. Figure 11(c) shows the comparison of the minimum required $\mathcal{P}$ with all $C_{max}$ evaluated $10^9$ randomly generated request frames and $E$ = 1. It illustrates that $LS$ algorithm achieves the lowest $\mathcal{P}$.

### 4.4.2 Non-uniform Traffic

Figure 12(a) and (b) illustrate the distribution of $H$ and $C_{max}$ of Log$_2$(128,14) under the non-uniform traffic model, respectively. The minimum required $\mathcal{P}$ is described in Fig. 12(c). The load balancing capability under the non-uniform traffic is very similar to the uniform traffic case. It indicates that the load balancing capability of each algorithm is independent of traffic pattern. It is notable that since all the algorithms in $g_H$ are very similar in terms of BP and hardware cost, the $LS$ algorithm is the best routing algorithm in $g_H$ under both of traffic models.

## 4.5 Algorithm Complexity

The complexities of all routing algorithms are summarized in Table 2. Since the worst case scenario of plane selecting in Log$_2$($N$, $m$) is to try all $m$ planes for a request, thus,

**Table 2** Complexity of routing algorithm for $\mathrm{Log}_2(N,m)$.

| Algorithm | Complexity of Algorithm |
|---|---|
| Danilewicz's algorithm ($D$) | $O(m \cdot N^2 log_2 N)$ |
| Otherwise | $O(m \cdot log_2 N)$ |

the complexity of plane selecting is $O(m)$. We need also $O(\log_2 N)$ time to check the path availability for each request, the overall time complexity of all algorithms, except $D$ algorithm [6], is just $O(m{\cdot}\log_2 N)$. For the $D$ algorithm, its complexity is as high as $O(m \cdot N^2 log_2 N)$ since it needs to calculate several complex matrices to achieve its low BP feature.

## 5. Experimental Results for Circuit Switching

Optical circuit switching (OCS for short) enables the switching of long-lived bulk optical data. For any data transmission in the OCS networks, connection paths between a source and a destination node have to be established before the data is transmitted. Therefore, the OCS guarantees the amount of bandwidth which is reserved to each connection and available to the connection all the time. On the reservation process in each switch, desired path is allocated for a connection when the switch can satisfy the demand of the connection, otherwise the request will be refused. The data transmission does not commence until the request succeed to reserve all of the switch nodes on the connection path.

In the OCS, no connection bandwidth can be shared by other connections. To construct the OCS based multi-$\log_2 N$ switches, the lower hardware cost is more important rather than the faster routing. As mentioned in the previous OPS simulation, the load balance is also important to reduce the peak power demand. In this section, routing algorithms are compared for several switching performances: blocking probability, hardware cost and load distribution (for complexity, see Table 2), through the circuit switch simulation according to [15].

### 5.1 Simulation Setting and Parameters

In this simulation, we assume that a connection request has an exponential holding time and arrives at each input port according to Poisson process independently, and an arriving request is destined to any output port based on the specified traffic model. Our OPS simulation program consists of the request frame generator and eight request routers for eight routing algorithms: The request frame generator generates a request frame when a connection request arrives at each input port with arrival rate, say $\lambda$. Each request router applies the corresponding routing algorithm to route each request in the frame step by step sequentially into the router. If a request is blocked at all the planes, the request is refused in the OCS. Each request has the exponential holding time with mean $1/\mu$. When the holding time of the request is finished, each request router also disconnects the expired requests from the routers. In this paper, we define $\rho(=\lambda/\mu)$ as the work load for input ports. The simulation procedure and parameters are shown as follows:

Simulation procedure of the circuit switch

```
{Each algorithm has own request router, and it is initialized only once through the simulation}
repeat
    Request frame generator generates a request frame F according to λ on the specified traffic model
    Set holding time for each request in F according to μ
    Determine order of requests randomly
    for each routing algorithm do
        for Each request in F do
            A request is set up by the request router based on the specified routing algorithm
            if a request is blocked then
                Count blocking
            end if
        end for
        Record C_max and C_min
        Disconnect expired requests by the request router
    end for
until Given #Requests are generated
return simulation results
```

Simulation parameters

| | |
|---|---|
| Network size($N$) | 8, 16, 32, 64, 128, 256 |
| #planes ($m$) | 1,2,. . ., until blocking probability $\leq 10^{-6}$ |
| Work Load ($\rho$) | 0.5, 0.6, 0.7, 0.8, 0.9, 1.0 |
| #Requests | $10^7$ |
| $C_{max}$ ($C_{min}$) | Maximum (minimum) plane load |

Simulation program was implemented by C on a cluster workstation — Opteron 2.0 GHz cluster.

### 5.2 Blocking Probability

In the OCS simulation, we don't adopt the *tagged request* based simulation as the previous OPS simulation, because the request frame is dynamically generated. Therefore, we calculate the connection blocking probability (BP) as follows.

$$BP = \frac{\text{\#blocked acceptable requests}}{\text{\#all the acceptable requests}}, \tag{5}$$

where the *acceptable request* denotes the request which is not blocked by other requests at an input port/output port.

#### 5.2.1 Uniform Traffic

Figure 13 and Fig. 14 illustrate results of BP versus number of planes for $\mathrm{Log}_2(128,m)$ and $\mathrm{Log}_2(256,m)$ with $\rho = 1.0$, respectively. We can also roughly divide routing algorithms into three groups, such as $g_H$, $g_M$ and $g_L$[†], same as previous OPS simulation. The BPs of above figures are smaller than the OPS simulation, since some requests could be blocked at input and output ports.

Figure 15 and Fig. 16 illustrate BPs versus work load ($0.5 \leq \rho \leq 1.0$) for $\mathrm{Log}_2(128,6)$ and $\mathrm{Log}_2(256,6)$, respectively. The BP of each algorithm in the same group is very close to each other. As noted in the OPS simulation, the

[†] $g_H$=$\{LS, R, CD, CS\}$, $g_M$=$\{STU\}$, $g_L$=$\{MI, P, D\}$

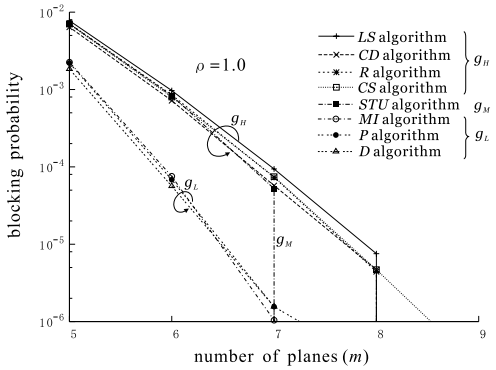



**Fig. 13** Blocking probability on Log$_2$(128,$m$) under the uniform traffic.

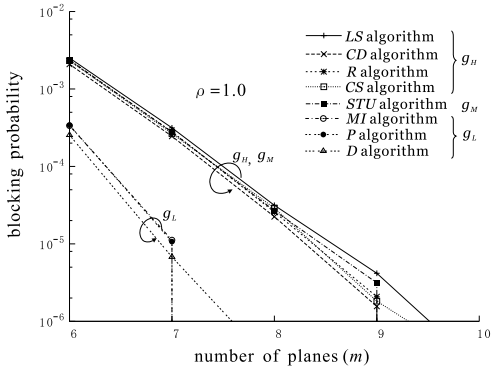



**Fig. 14** Blocking probability on Log$_2$(256,$m$) under the uniform traffic.

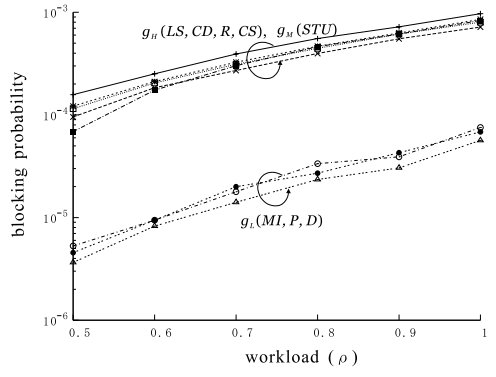



**Fig. 15** Blocking probability versus the work loads on Log$_2$(128,6) under the uniform traffic.

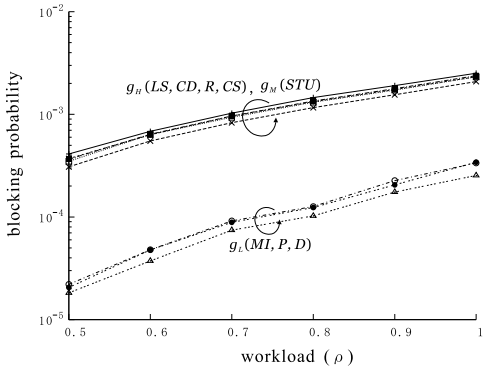



**Fig. 16** Blocking probability versus the work loads on Log$_2$(256,6) under the uniform traffic.

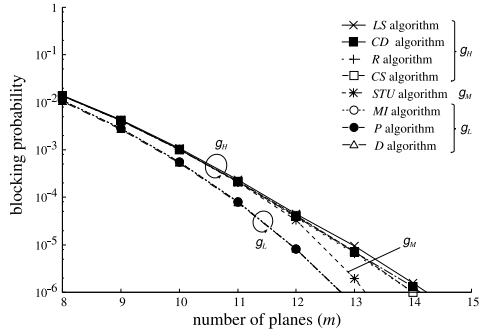



**Fig. 17** Blocking probability on Log$_2$(256, $m$) under the non-uniform traffic.

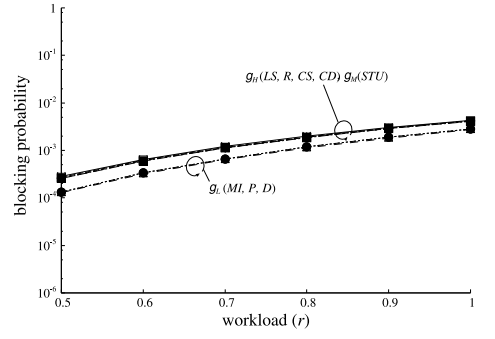



**Fig. 18** Blocking probability vs. work load $\rho$ on Log$_2$(256, 9) under the non-uniform traffic.

$STU$ algorithm is sensitive to the number of planes, and some requests remain on several planes due to their holding time in the OCS simulation. Therefore, the BP of $STU$ algorithm becomes close to $g_H$ even as the number of planes are increased.

### 5.2.2 Non-uniform Traffic

Figure 17 and Fig. 18 describe the BPs under the non-uniform traffic case. We can see that BPs of all routing algorithms close to each other. The main reason of the similar BPs is that the blocking paths are further increased by not only the traffic pattern as discussed in the OPS simulation but also remained connections in the request router.

## 5.3 Hardware Cost (Number of Switch Elements)

For the blocking design multi-log$_2N$ switches, the required planes can be reduced when the negligible BP is allowed. We assume blocking probability less than $10^{-6}$ as negligible here as the previous OPS simulation and compare the minimum required number of DCs under this assumption. The comparison under the uniform and the non-uniform traffic case are shown in Fig. 19 and Fig. 20, respectively. In the OCS simulation, the hardware cost among routing algo-

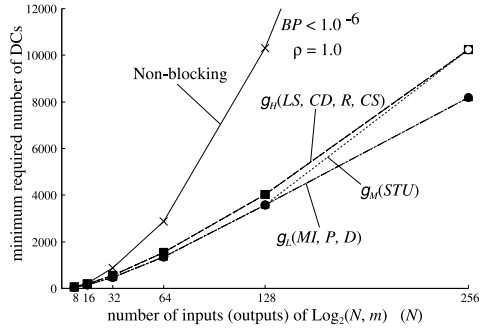



**Fig. 19** Minimum number of planes with blocking probability $< 1.0^{-6}$ on $\rho = 1.0$, under uniform traffic model.

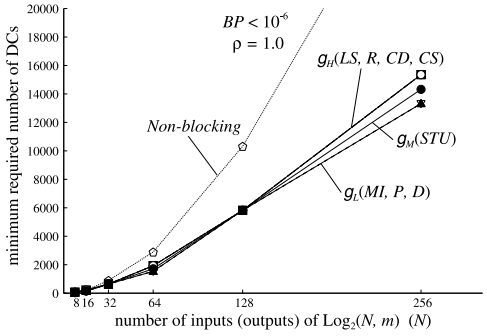



**Fig. 20** Minimum number of planes with blocking probability $< 1.0^{-6}$ on $\rho = 1.0$, under non-uniform traffic model.

rithms is very similar to each other. For reference, we also include the required number of DCs for the non-blocking condition.

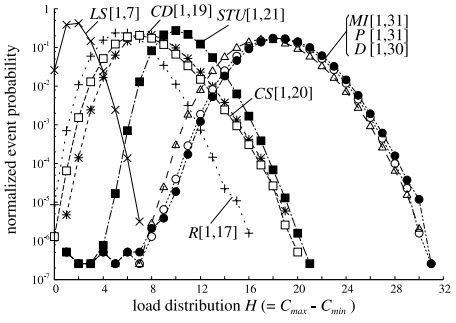
We can observe that the $g_L$ achieve the lowest hardware cost. Although the hardware cost of $g_H$ is always higher than others, there is no big difference among them.

In the OCS simulation under the uniform traffic case, the $g_M$ follows $g_L$ when the network size is less than 128, but follows $g_H$ when $N = 256$. Since the difference of minimum required hardware cost is very close to each other when the network size is less than 128, we calculated the minimum hardware cost according to non-negligible BP. It can be seen that when $N \geq 256$, the $STU$ algorithm lose its saving plane capability due to the increase of remained requests in the request router.

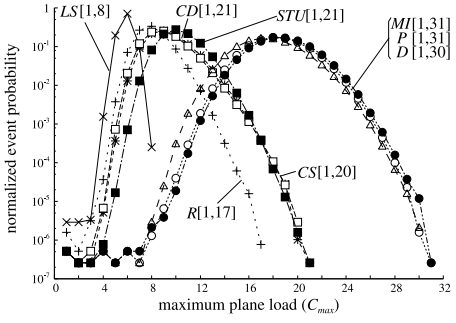
### 5.4 Load Balance and Peak Power Requirement

As mentioned earlier, a heavy plane load on the multi-$\log_2 N$ switch is critical, since it lead to high-capacity power equipment for the maximum peak power and a large number of disconnection on a plane due to a failure. Therefore, requests need to be distributed as uniform as possible to network planes.

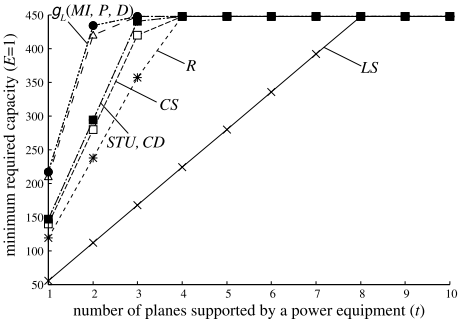
To compare the load balancing capability, although we use the same notations as previous OPS simulation, such as $H$, $C_{max}$, $C_{min}$, and $[a,b]$, $C_{max}$ and $C_{min}$ of all algorithms are evaluated for randomly generated $10^7$ request frames be-




(a) Distribution of the plane load balance index $H$.




(b) Distribution of the maximum plane load $C_{max}$.




(c) Minimum required capacity of power equipment.

**Fig. 21** Load balancing on $Log_2(128,10)$ under the uniform traffic, $\rho = 1.0$.

cause frequent switch control is usually not required. The peak power requirement $\mathcal{P}$ is also calculated by Eq. (4).

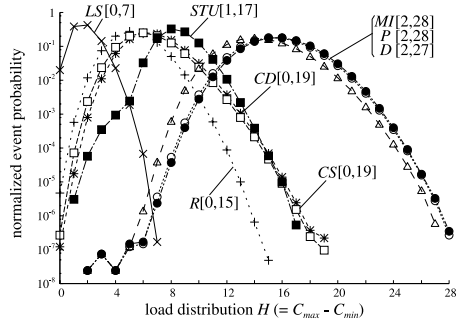
#### 5.4.1 Uniform Traffic

Figure 21(a) and (b) describe the normalized distribution of $H$ and of all the $C_{max}$ through this simulation for each routing algorithm on $Log_2(128,8)$ with $\rho = 1.0$, respectively. Although the OCS simulation is different from the OPS due to holding times of packets, the proposed $LS$ algorithm still keep the better load balancing than others.

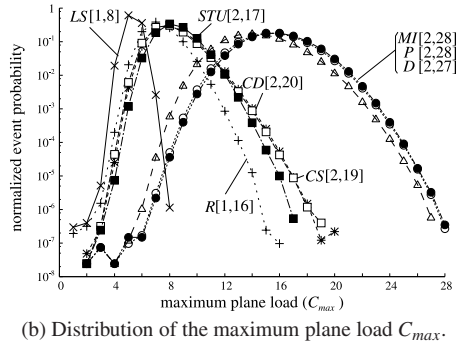
In addition, it is interesting to see that the $C_{max}$ distribution of the $R$ algorithm is better than the $CD$ algorithm in the OCS as contrasted to the OPS simulation. In the OPS, it can be seen that the $CD$ algorithm provides the better load balancing due to selecting plane based on the fixed order rotation. This selecting manner equally provides setup chances to all planes. In contrast, $R$ algorithm provides setup chances based on random selecting. In the OCS, several connections

may be remained in the request router due to their holding time. Since no factor can change the selecting order of $CD$ algorithm, the connections remained strongly increase the plane load. In contrast, random selecting strategy of $R$ algorithm is not affected so much by such connections.

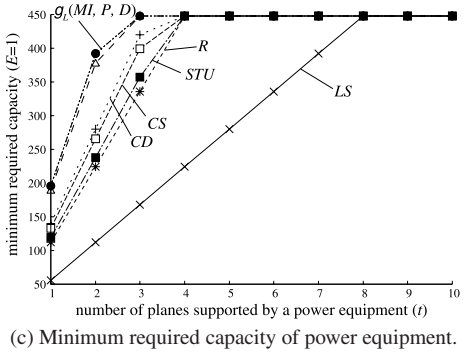
We also show the minimum capacity of the minimum required $\mathcal{P}$ calculated by Eq. (4) shown in Fig. 21(c). It can be seen that $LS$ algorithm can significantly reduce $\mathcal{P}$.




(a) Distribution of the plane load balance index $H$.




(b) Distribution of the maximum plane load $C_{max}$.




(c) Minimum required capacity of power equipment.

**Fig. 22** Load balancing on Log$_2$(128,12) under the non-uniform traffic, $\rho = 1.0$.

### 5.4.2 Non-uniform Traffic

Figure 22(a) and (b) plot normalized distributions of all $H$ and $C_{max}$ under the non-uniform traffic simulation. This results are very similar to results of the uniform traffic case.

Figure 22(c) illustrates the comparison of the minimum required capacity of a power equipment calculated from results shown in Fig. 22(b). Since the peak power is very similar to that under the uniform traffic case, the distribution of $C_{max}$ and $H$ are also similar. From the above observations, we can understand that $LS$ algorithm significantly affects the peak power and the load balancing capability in OCS is independent of both traffic models.

## 6. Conclusion and Remarks

The blocking design of multi-log$_2N$ networks is attractive to reduce their hardware cost. In this paper, we fully compared the performances of the seven available routing algorithms and also one newly proposed algorithm for multi-log$_2N$ networks by using packet switching and circuit switching techniques under both the uniform and the non-uniform traffic models. All the comparisons are summarized in Table 3. As measures of switching performance, we considered blocking probability, complexity and load balancing capability. As results, we found that the routing algorithms which try to pack requests into few number of planes usually achieve a lower blocking probability, but such algorithms require high peak power supply and also advanced power dissipation. In contrast, our newly proposed routing algorithm provide a good load-balancing capability but may result in a higher blocking probability. We also found that the load balancing capability is independent of both traffic models. Thus, how to design a routing algorithm to achieve a nice trade-off between a low blocking probability and a good load-balancing capability is an interesting future work.

## Acknowledgement

This work was supported in part by Japan Science Promotion Society (JSPS) under Grant-In-Aid of Scientific Research (C) 19500050 and (B) 20300022.

### References

[1] C.T. Lea, "Multi-log$_2N$ networks and their applications in high-speed electronic and photonic switching systems," IEEE Trans. Commun., vol.38, no.10, pp.1740–1749, Oct. 1990.

**Table 3** Summarized performance comparison of routing algorithms.

| | | Blocking Probability | Hardware Cost | Complexity | Load Balancing capability |
|---|---|---|---|---|---|
| Packet switch | uniform traffic | $g_L < g_M < g_H$ | $g_L < g_M, g_H$ | $others \ll D$ | $g_L < g_M < CS < R < CD < LS$ |
| | non-uniform traffic | $g_L < g_M, g_H$ | | | |
| Circuit switch | uniform traffic | $g_L < g_M \leq g_H$ | $g_L < g_M \leq g_H$ | | $g_L < g_M < CS < CD < R < LS$ |
| | non-uniform traffic | $g_L \leq g_M \leq g_H$ | $g_L < g_M < g_H$ | | |

$g_H = \{LS, R, CD, CS\}$, $g_M = \{STU\}$, $g_H = \{MI, P, D\}$

[2] F.H. Chang, J.Y. Guo, and F.K. Hwang, "Wide-sense nonblocking for multi-$log_dN$ networks under various routing strategies," Theor. Comput. Sci., vol.352, pp.232–239, 2006.
[3] X. Jiang, P.H. Ho, and S. Horiguchi, "Performance modeling for all-optical photonic switches based on the vertical stacking of banyan network structures," IEEE J. Sel. Areas Commun., vol.23, no.8, pp.1620–1631, Aug. 2005.
[4] X. Jiang, H. Shen, M.R. Khandker, and S. Horiguchi, "Blocking behaviors of crosstalk-free optical banyan networks on vertical stacking," IEEE/ACM Trans. Netw., vol.11, no.6, pp.982–993, Dec. 2003.
[5] C.T. Lea and D.J. Shyy, "Tradeoff of horizontal decomposition versus vertical stacking in rearrangeable nonblocking networks," IEEE Trans. Commun., vol.39, no.6, pp.899–904, June 1991.
[6] G. Danilewicz, W. Kabacinski, M. Michalski, and M. Zal, "Wide-sense nonblocking multiplane photonic banyan-type switching fabrics with zero crosstalk," Proc. IEEE ICC 2006, pp.11–15, Istanbul, Turkey, June 2006.
[7] M.M. Vaez and C. Lea, "Strictly nonblocking directional-coupler-based switching networks under crosstalk constraint," IEEE Trans. Commun., vol.48, no.2, pp.316–323, Feb. 2000.
[8] G. Maier and A. Pattavina, "Design of photonic rearrangeable networks with zero first-order switching-element-crosstalk," IEEE Trans. Commun., vol.49, no.7, pp.1268–1279, July 2001.
[9] W. Kabacinski, G. Danilewicz, and M. Glabowski, "SWITCH FABRIC CONTROL," Optical Switching, pp.307–332, Springer, 2005.
[10] M.H. Ackroyd, "Call repacking in connecting networks," IEEE Trans. Commun., vol.COM-27, no.3, pp.589–591, March 1979.
[11] A. Jajszczyk and G. Jekel, "A new concept — Repackable networks," IEEE Trans. Commun., vol.41, no.8, pp.1232–1237, Aug. 1993.
[12] Y. Mun, Y. Tang, and V. Devarajan, "Analysis of call packing and rearrangement in a multi stage switch," IEEE Trans. Commun., vol.42, no.2/3/4, pp.252–254, Feb./March/April 1994.
[13] Y. Yang and J. Wang, "Wide-sense nonblocking clos networks under packing strategy," IEEE Trans. Comput., vol.48, no.3, pp.265–284, March, 1999.
[14] G.I. Papadimitriou, C. Papazoglou, and A.S. Pomportsis, "Optical switching: Switch fabrics, techniques, and architectures," J. Lightwave Technol., vol.21, no.2, pp.384–405, Feb. 2003.
[15] A. Pattavina, Switching Theory Architectures and Performance in Broadband ATM Networks, John Wiley & Sons, 1998.

**Yusuke Fukushima** received the B.S. degree from Department of Informatics of Yamagata University in 2004. He received the M.S. degree from Graduate School of Information Sciences, Tohoku University in 2006. He is currently studing for Ph.D. degree. His interests include interconnection networks and distributed systems.

**Xiaohong Jiang** received his B.S., M.S. and Ph.D. degrees in 1989, 1992, and 1999 respectively, all from Xidian University, Xi'an, China. He is currently an Associate Professor in the Department of Computer Science, Graduate School of Information Science, TOHOKU University, Japan. Before joining TOHOKU University, Dr.Jiang was an assistant professor in the Graduate School of Information Science, Japan Advanced Institute of Science and Technology (JAIST), from Oct. 2001 to Jan. 2005. Dr. Jiang was a JSPS (Japan Society for the Promotion of Science) post-doctoral research fellow at JAIST from Oct. 1999–Oct. 2001. He was a research associate in the Department of Electronics and Electrical Engineering, the University of Edinburgh from Mar.1999–Oct.1999. Dr. Jiang's research interests include optical switching networks, routers, network coding, WDM networks, VoIP, interconnection networks, IC yield modeling, timing analysis of digital circuits, clock distribution and fault-tolerant technologies for VLSI/WSI. He has published over 120 referred technical papers in these areas. He is a member of IEEE.

**Susumu Horiguchi** (M'81–SM'95) received the B.Eng. the M.Eng. and Ph.D. degrees from Tohoku University in 1976, 1978 and 1981 respectively. He is currently a Full Professor in the Graduate School of Information Sciences, Tohoku University. He was a visiting scientist at the IBM Thomas J. Watson Research Center from 1986 to 1987. He was also a professor in the Graduate School of Information Science, JAIST (Japan Advanced Institute of Science and Technology). He has been involved in organizing international workshops, symposia and conferences sponsored by the IEEE, IEICE, IASTED and IPS. He has published over 150 papers technical papers on optical networks, interconnection networks, parallel algorithms, high performance computer architectures and VLSI/WSI architectures. Prof. Horiguchi is members of IPS and IASTED.