A Linear Programming Duality Approach to Analyzing Strictly Nonblocking d-ary Multilog Networks under General Crosstalk Constraints

Hung Q. Ngo, Anh Le, and Yang Wang
Computer Science and Engineering,
201 Bell Hall,
State University of New York at Buffalo,
Amherst, NY 14260, USA.
{hungngo,yw43,anhle}@cse.buffalo.edu

Abstract

When a switching network topology is used for constructing optical cross-connects, as in the circuit switching case, no two routes are allowed to share a link. However, if two routes share too many switching elements, then crosstalk introduced at those switching elements degrades signal quality. Vaez and Lea [20] introduced a parameter c which is the maximum number of distinct switching elements a route can share with other routes in the network. This is called the general crosstalk constraint. This paper presents a new method of analyzing strictly nonblocking multi-log networks under this general crosstalk constraint using linear programming duality.

We improve known results on several fronts: (a) our sufficient conditions are better than known sufficient conditions for $\log_d(N, 0, m)$ to be strictly nonblocking under general crosstalk constraints, (b) our results are on d-ary multi-log networks while known results are on binary networks, and (c) for several ranges of the parameter c, we give the first known necessary conditions for this problem which also match our sufficient conditions from the LP-duality approach.

One important advantage of the LP-duality approach is the ease and brevity of sufficiency proofs. All one has to do is to verify that a solution is indeed dual-feasible and the dual-objective value automatically gives us a sufficient condition. Earlier works on this problem relied on combinatorial arguments which are quite intricate and somewhat error-prone.

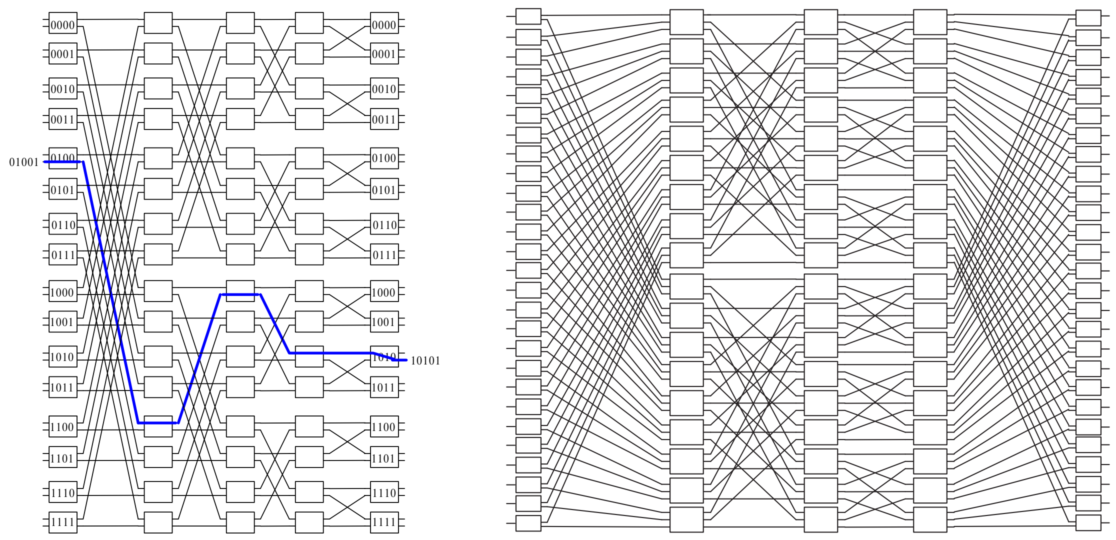
Keywords: strictly nonblocking, optical switches, d-ary multi-log switching networks, general crosstalk constraint, crosstalk-free, linear programming duality.

1 Introduction

We study the general d-ary multi-log switch architecture with multiple vertically stacked inverse Banyan planes $\mathrm{BY}^{-1}(n)$, as illustrated in Figure 1. The d-ary multi-log switches have been attractive for both electronic and photonic domains, because they have small depth ($\log N$), self-routing capability, absolute signal loss uniformity, and good fault tolerance [6, 9, 11–13, 17, 20]. Hereafter, we use $\log_d(N, 0, m)$ to denote a d-ary multi-log switch with m vertically stacked inverse Banyan planes.

There are three levels of nonblockingness typically studied in the switching network literature: rearrangeably nonblocking (RNB), wide-sense nonblocking (WSNB), and strictly nonblocking (SNB). The reader is referred to [7, 15] for their precise definitions. This paper focuses on analyzing the SNB $\log_d(N, 0, m)$ network.

When the multi-log architecture is used to design a photonic switch, each switching element (SE) needs to be replaced by a functionally equivalent optical component. For instance, when $d = 2$ we



(a) The inverse Banyan network $\mathrm{BY}^{-1}(5)$ (b) The $\log_3(27, 0, 2)$ network

Figure 1: Illustration of the Multi-log Network

can use directional couplers as SEs [14, 18, 22]. However, directional couplers and many other optical switching elements suffer from optical crosstalk between interfering channels, which is one of the major obstacles in designing cost-effective switches [2, 3, 19]. To cope with crosstalk, the *general crosstalk constraint* was introduced by Vaez and Lea [20], which forces each new route to share at most c SEs with other active routes in the switch, where c is a parameter of the design. When $c = 0$, no two routes are allowed to share a common SE, and we have a *crosstalk-free* switch [2, 3, 9, 11, 13, 20, 21]. On the other hand, when $c = n = \log_d N$ a route can share any number of SEs with other routes (but no shared link), because each route in a $\log_d(N, 0, m)$ network contains exactly n SEs. In this case only link-blocking takes effect, and it is commonly referred to as the *link-blocking case*, which is relevant to electronic switches [1, 4–8, 10, 12, 16, 17]. We shall refer to a switching network which is strictly nonblocking (SNB) under this general crosstalk constraint a *c-SNB network*.

Prior to this paper, the only known results are sufficient conditions for the binary $\log_2(N, 0, m)$ network to be c-SNB [20]. Specifically, let $N = 2^n$ Vaez and Lea showed that, when $n = 2k + 1$

$$m \geq \begin{cases} 2^{k+1} + \left\lfloor \frac{2^{k+1}}{c+2} \right\rfloor - 1 & 0 < c \leq k \\ 2^{k+1} + \left\lfloor \frac{2^k}{c+2} \right\rfloor & k < c \leq 2k \end{cases} \tag{1}$$

is sufficient for $\log_2(N, 0, m)$ to be c-SNB. When, $n = 2k$ the sufficient condition is

$$m \geq \begin{cases} 2^k + 2^{k-1} + \left\lfloor \frac{2^k}{c+1} \right\rfloor - 1 & 0 < c \leq k \\ 2^k + 2^{k-1} - 1 & k < c \leq 2k - 1 \end{cases} \tag{2}$$

Our main contributions are as follows. We present a new method of analyzing strictly nonblocking multi-log networks under general crosstalk constraint using linear programming duality. Our approach improves previous results from several fronts: (a) our sufficient conditions are better than known sufficient conditions (1) and (2) for $\log_d(N, 0, m)$ to be c-SNB, (b) our results are on d-ary multi-log networks while known results are on binary networks, (c) for several ranges of the parameter c, we give the first known necessary conditions for this problem which also match our sufficient conditions from the LP-duality approach.

One important advantage of the LP-duality approach is the ease and brevity of sufficiency proofs. All we have to do is to check that a solution is indeed dual-feasible and the dual-objective value automatically gives us a sufficient condition. Earlier works on this problem relied on combinatorial arguments which are quite intricate and somewhat error-prone.

The rest of the paper is organized as follows. Section 2 establishes basic notations and presents a simple algebraic view of $\log_d(N, 0, m)$ networks, which are used throughout the paper. Section 3 presents the LP-duality approach and sufficient conditions for the $\log_d(N, 0, m)$ network to be c-SNB. Section 4 presents the necessary conditions for several ranges of c which match the sufficiency conditions.

2 Preliminaries

We first establish notations which will be used throughout the paper. For any positive integers l, d, let $[l]$ denote the set $\{1, \dots, l\}$, $\mathbb{Z}_d$ denote the set $\{0, \dots, d-1\}$ which can be thought of as d-ary "symbols," $\mathbb{Z}_d^l$ denote the set of all d-ary strings of length l, b^l denote the string with symbol $b \in \mathbb{Z}_d$ repeated l times (e.g., $3^4 = 3333$), $|\mathbf{s}|$ denote the length of any d-ary string $\mathbf{s}$ (e.g., $|31| = 2$), $\mathbf{s}_{i..j}$ denote the substring $s_i \cdots s_j$ of a string $\mathbf{s} = s_1 \dots s_l \in \mathbb{Z}_d^l$. If $i > j$ then $\mathbf{s}_{i..j}$ is the empty string.

Let $N = d^n$. We consider the $\log_d(N, 0, m)$ network, which denotes the stacking of m copies of the d-ary inverse Banyan network $\text{BY}^{-1}(n)$ with N inputs and N outputs. Each of these copies is called a *BY*$^{-1}(n)$*-plane* (Figure 1b illustrates a multi-log network with two planes). We label the inputs and outputs of a $\text{BY}^{-1}(n)$-plane with d-ary strings of length n. Specifically, each input $\mathbf{x} \in \mathbb{Z}_d^n$ and output $\mathbf{y} \in \mathbb{Z}_d^n$ have the form $\mathbf{x} = x_1 \cdots x_n$, $\mathbf{y} = y_1 \cdots y_n$, where $x_i, y_i \in \mathbb{Z}_d$, $\forall i \in [n]$.

Also, label the $d \times d$ SEs in each of the n stages of a $\text{BY}^{-1}(n)$-plane with d-ary strings of length $n-1$. An input $\mathbf{x}$ (resp. output $\mathbf{y}$) is connected to the SE labeled $\mathbf{x}_{1..n-1}$ in the first stage (resp. $\mathbf{y}_{1..n-1}$ in the last stage).

For the sake of clarity, let us first consider a small example. Consider the unicast request $(\mathbf{x}, \mathbf{y}) = (01001, 10101)$ when $d = 2, n = 5$. The input $\mathbf{x} = 01001$ is connected to the SE labeled 0100 in the first stage, which is connected to two SEs labeled 0100 and 1100 in the second stage, and so on. The unique path from $\mathbf{x}$ to $\mathbf{y}$ in the $\text{BY}^{-1}(n)$-plane can be explicitly written out (see Figure 1a):

input $\mathbf{x}$	01001
stage-1 SE	0100
stage-2 SE	**1**100
stage-3 SE	**10**10
stage-4 SE	**101**0
stage-5 SE	**1010**
output $\mathbf{y}$	10101

We can see clearly the pattern: the prefixes of $\mathbf{y}_{1..n-1}$ are "taking over" the prefixes of $\mathbf{x}_{1..n-1}$ on the path from $\mathbf{x}$ to $\mathbf{y}$. In general, the unique path $R(\mathbf{x}, \mathbf{y})$ in a $\text{BY}^{-1}(n)$-plane from an arbitrary input $\mathbf{x}$ to an arbitrary output $\mathbf{y}$ is exactly the following:

input $\mathbf{x}$	$x_1 x_2 \dots x_{n-1} x_n$
stage-1 SE	$x_1 x_2 \dots x_{n-1}$
stage-2 SE	$y_1 x_2 \dots x_{n-1}$
stage-3 SE	$y_1 y_2 \dots x_{n-1}$
$\vdots$	$\vdots$
stage-n SE	$y_1 y_2 \dots y_{n-1}$
output $\mathbf{y}$	$y_1 y_2 \dots y_{n-1} y_n$

Now, consider two unicast requests $(\mathbf{a}, \mathbf{b})$ and $(\mathbf{x}, \mathbf{y})$. From the observation above, on the same $\text{BY}^{-1}(n)$-plane the two routes $R(\mathbf{a}, \mathbf{b})$ and $R(\mathbf{x}, \mathbf{y})$ share a SE (also called a node) if and only if there is some $j \in [n]$ such that $b_{1..j-1} = y_{1..j-1}$ and $a_{j..n-1} = x_{j..n-1}$. In this case, the two paths intersect at a stage-j SE. It should be noted that two requests' paths may intersect at more than one SE.

For any two d-ary strings $\mathbf{u}, \mathbf{v} \in \mathbb{Z}_d^l$, let $\text{PRE}(\mathbf{u}, \mathbf{v})$ denote the *longest common prefix*, and $\text{SUF}(\mathbf{u}, \mathbf{v})$ denote the *longest common suffix* of $\mathbf{u}$ and $\mathbf{v}$, respectively. For example, if $\mathbf{u} = 0100110$ and $\mathbf{v} = 0101010$, then $\text{PRE}(\mathbf{u}, \mathbf{v}) = 010$ and $\text{SUF}(\mathbf{u}, \mathbf{v}) = 10$. The following propositions immediately follow.

Proposition 2.1. *Let $(\mathbf{a}, \mathbf{b})$ and $(\mathbf{x}, \mathbf{y})$ be two unicast requests. Then their corresponding routes $R(\mathbf{a}, \mathbf{b})$ and $R(\mathbf{x}, \mathbf{y})$ in a $BY^{-1}(n)$-plane share at least a common SE if and only if*

$$|\text{SUF}(\mathbf{a}_{1..n-1}, \mathbf{x}_{1..n-1})| + |\text{PRE}(\mathbf{b}_{1..n-1}, \mathbf{y}_{1..n-1})| \geq n - 1. \tag{3}$$

Moreover, the routes $R(\mathbf{a}, \mathbf{b})$ and $R(\mathbf{x}, \mathbf{y})$ intersect at exactly one SE if and only if

$$|\text{SUF}(\mathbf{a}_{1..n-1}, \mathbf{x}_{1..n-1})| + |\text{PRE}(\mathbf{b}_{1..n-1}, \mathbf{y}_{1..n-1})| = n - 1, \tag{4}$$

in which case the common SE is an SE at stage $|\text{PRE}(\mathbf{b}_{1..n-1}, \mathbf{y}_{1..n-1})| + 1$ of the $BY^{-1}(n)$-plane.

Proposition 2.2. *Let $(\mathbf{a}, \mathbf{b})$ and $(\mathbf{x}, \mathbf{y})$ be two unicast requests. Then their corresponding routes $R(\mathbf{a}, \mathbf{b})$ and $R(\mathbf{x}, \mathbf{y})$ in a $BY^{-1}(n)$-plane share at least a common link if and only if*

$$|\text{SUF}(\mathbf{a}_{1..n-1}, \mathbf{x}_{1..n-1})| + |\text{PRE}(\mathbf{b}_{1..n-1}, \mathbf{y}_{1..n-1})| \geq n. \tag{5}$$

3 Sufficient Conditions

In Section 3.1, we show that the number of planes blocking a new request is the objective value of a general linear program which is dependent on the parameters of the problem (c, d, n), but independent from the request under consideration. By weak duality, in order to bound the number of planes blocking any new request we can just use the objective value of any dual-feasible solution.

In Sections 3.2 and 3.3, we construct families of solutions which are dual-feasible. Then, depending of the relationships between the problem's parameters (c, d, n), we choose the best dual-feasible solution from the constructed families for bounding the number of blocking planes.

As we have mentioned earlier, the strength of our method is in the brevity of the sufficiency proofs. However, this advanrage has a cost hidden from the readers. Constructing these families of dual-feasible solutions involves a good amount of experimentation, educated guesses, and Maple/Matlab coding.

3.1 The Linear Programming Duality Approach

Let $(\mathbf{a}, \mathbf{b})$ be an arbitrary request. For each $i \in \{0, \dots, n-1\}$, let A_i be the set of inputs $\mathbf{x}$ other than $\mathbf{a}$, where $\mathbf{x}_{1..n-1}$ shares a *suffix* of length exactly i with $\mathbf{a}_{1..n-1}$. Formally, define

$$A_i := \{\mathbf{x} \in \mathbb{Z}_d^n - \{\mathbf{a}\} \mid \text{SUF}(\mathbf{x}_{1..n-1}, \mathbf{a}_{1..n-1}) = i\}.$$

Similarly, for each $j \in \{0, \dots, n-1\}$, let B_j be the set of outputs $\mathbf{y}$ other than $\mathbf{b}$ which share a *prefix* of length exactly j with $\mathbf{b}$, namely

$$B_j := \{\mathbf{y} \in \mathbb{Z}_d^n - \{\mathbf{b}\} \mid \text{PRE}(\mathbf{y}_{1..n-1}, \mathbf{b}_{1..n-1}) = j\}.$$

Note that $|A_i| = |B_i| = d^{n-i} - d^{n-1-i}$, for all $0 \leq i \leq n-1$. Suppose the network $\log_d(N, 0, m)$ already had some routes established. Consider a $\text{BY}^{-1}(n)$-plane which blocks the new request $(\mathbf{a}, \mathbf{b})$ (i.e., $(\mathbf{a}, \mathbf{b})$ cannot be routed through the plane). There can only be three cases for which this happens:

Case 1: There is a request $(\mathbf{x}, \mathbf{y})$ routed in the plane for which $R(\mathbf{x}, \mathbf{y})$ and $R(\mathbf{a}, \mathbf{b})$ share a link. By Proposition 2.2, it must be the case that $(\mathbf{x}, \mathbf{y}) \in A_i \times B_j$ for some $i + j \geq n$. We will refer to $(\mathbf{x}, \mathbf{y})$ as a *link-blocking request* with respect to $(\mathbf{a}, \mathbf{b})$. We will drop the qualifier "with respect to" when it is unambiguous from the current context which request is being blocked.

Case 2: There is a request $(\mathbf{x}, \mathbf{y})$ whose route $R(\mathbf{x}, \mathbf{y})$ already intersects c other routes at c distinct SEs on the same plane, and adding $(\mathbf{a}, \mathbf{b})$ would introduce an additional intersecting SE to the route $R(\mathbf{x}, \mathbf{y})$. We will refer to these c other requests as *secondary requests* accompanying $(\mathbf{x}, \mathbf{y})$, and refer to $(\mathbf{x}, \mathbf{y})$ as a *node-blocking request of type* 1.

In particular, by Proposition 2.1 we must have $(\mathbf{x}, \mathbf{y}) \in A_i \times B_j$ for some $i + j = n - 1$. Note that the common SE between $R(\mathbf{a}, \mathbf{b})$ and $R(\mathbf{x}, \mathbf{y})$ is at stage $j + 1$. The routes for secondary requests must thus intersect $R(\mathbf{x}, \mathbf{y})$ at stages strictly less than $j + 1$ or strictly greater than $j + 1$.

If a secondary request $(\mathbf{u}, \mathbf{v})$ has its route intersects $R(\mathbf{x}, \mathbf{y})$ at stage s with $1 \leq s < j + 1$, then it follows that $\text{PRE}(\mathbf{y}, \mathbf{v}) = s - 1 < j$, and $\text{SUF}(\mathbf{x}, \mathbf{u}) = n - 1 - (s - 1) = n - s > n - 1 - j = i$. Hence, $(\mathbf{u}, \mathbf{v}) \in A_i \times B_{s-1}$. We will refer to such $(\mathbf{u}, \mathbf{v})$ as a *left secondary request.*

If a secondary request $(\mathbf{u}, \mathbf{v})$ has its route intersects $R(\mathbf{x}, \mathbf{y})$ at stage s where $j + 1 < s \leq n$, then it follows that $\text{PRE}(\mathbf{y}, \mathbf{v}) = s - 1 > j$, and $\text{SUF}(\mathbf{x}, \mathbf{u}) = n - 1 - (s - 1) = n - s < n - 1 - j = i$. Hence, $(\mathbf{u}, \mathbf{v}) \in A_{n-s} \times B_j$. We will refer to such $(\mathbf{u}, \mathbf{v})$ as a *right secondary request.*

To summarize, there are two types of secondary requests accompanying $(\mathbf{x}, \mathbf{y})$: the *left secondary requests* are the requests $(\mathbf{u}, \mathbf{v}) \in A_i \times B_{j'}$ for some $j' < j$, and the *right secondary requests* are the requests $(\mathbf{u}, \mathbf{v}) \in A_{i'} \times B_j$ for some $i' < i$. For each $i' < i$ there is at most one right secondary request in $A_{i'} \times B_j$. Similarly, for each $j' < j$ there is at most one left secondary request in $A_i \times B_{j'}$.

Case 3: There are $c+1$ requests in the plane each of whose routes intersects $(\mathbf{a}, \mathbf{b})$ at exactly one SE. These will be called *node-blocking requests of type* 2. If $(\mathbf{x}, \mathbf{y})$ is such a request, then $(\mathbf{x}, \mathbf{y}) \in A_i \times B_j$ for some $i + j = n - 1$.

Theorem 3.1. *The number of blocking planes is the objective value of a feasible solution to the primal linear program as shown in Figure 2.*

Proof. Define the following variables. For each pair i, j such that $i + j \geq n$, let x_{ij} be the number of link-blocking requests in $A_i \times B_j$. For each pair i, j such that $i + j = n - 1$, let y_{ij} be the number of node-blocking requests of type 1, and z_{ij} be the number of node-blocking requests of type 2 in $A_i \times B_j$. For each pair i, j such that $i + j < n - 1$, let l_{ij} and r_{ij} be the number of left and right secondary requests in $A_i \times B_j$. The number of blocking planes is thus expressed in the objective function (6).

We next explain why the variables satisfy all the constraints. Recall that $|A_i| = d^{n-i} - d^{n-1-i}, \forall i$. Thus, the number of requests out of A_i is at most $d^{n-i} - d^{n-1-i}$, justifying constraint (7). Similarly, bounding the number of requests to B_j explains constraint (8). Constraint (9) expresses the fact that, for every node-blocking request of type 1 in $A_i \times B_j$ $(i+j = n-1)$, there must be c accompanying secondary requests (left or right). Lastly, for each node-blocking request of type 1 in $A_i \times B_j$ $(i + j = n - 1)$ constraint (10) says that for each $j' < j$ there is at most one left secondary request in $A_i \times B_{j'}$, and constraint (11) says that for each $i' < i$ there is at most one right secondary request in $A_{i'} \times B_j$. □

Remark 3.2. It should be noted that while the number of blocking planes is the objective value of some feasible solution to the primal LP, the converse may not hold; namely, a feasible solution to the primal LP (even if integral) may not give rise to a blocking configuration with the number of blocking planes equal to the objective value.

The Primal LP

Maximize

$$\sum_{i+j\geq n} x_{ij} + \sum_{i+j=n-1} y_{ij} + \frac{1}{c+1}\sum_{i+j=n-1} z_{ij} \tag{6}$$

Subject to

$$\begin{aligned}
\sum_{j:\, i+j\geq n} x_{ij} + \sum_{j:\, i+j=n-1} (y_{ij}+z_{ij}) + \sum_{j:\, i+j<n-1} (l_{ij}+r_{ij}) &\leq & d^{n-i} - d^{n-1-i} && \forall i && (7)\\
\sum_{i:\, i+j\geq n} x_{ij} + \sum_{i:\, i+j=n-1} (y_{ij}+z_{ij}) + \sum_{i:\, i+j<n-1} (l_{ij}+r_{ij}) &\leq & d^{n-j} - d^{n-1-j} && \forall j && (8)\\
cy_{ij} - \left(\sum_{i'<i} r_{i'j} + \sum_{j'<j} l_{ij'}\right) &= & 0 && i+j=n-1 && (9)\\
l_{ij'} - y_{ij} &\leq & 0 && i+j=n-1, j'<j && (10)\\
r_{i'j} - y_{ij} &\leq & 0 && i+j=n-1, i'<i && (11)\\
x_{ij}, y_{ij}, z_{ij}, l_{ij}, r_{ij} &\geq & 0 && \forall i,j && (12)
\end{aligned}$$

The Dual LP

Minimize

$$\sum_{i=0}^{n-1} (d^{n-i} - d^{n-1-i})u_i + \sum_{j=0}^{n-1} (d^{n-j} - d^{n-1-j})v_j \tag{13}$$

Subject to

$$\begin{aligned}
u_i + v_j &\geq & 1, && i+j\geq n && (14)\\
u_i + v_j + cw_{ij} - \sum_{i'<i} s_{i'j} - \sum_{j'<j} t_{ij'} &\geq & 1, && i+j=n-1 && (15)\\
u_i + v_j &\geq & \tfrac{1}{c+1}, && i+j=n-1 && (16)\\
u_i + v_j - w_{i,n-1-i} + t_{ij} &\geq & 0, && i+j<n-1 && (17)\\
u_i + v_j - w_{n-1-j,j} + s_{ij} &\geq & 0, && i+j<n-1 && (18)\\
u_i, v_j, s_{ij}, t_{ij} &\geq & 0, && \forall i,j && (19)
\end{aligned}$$

Figure 2: The Primal and the Dual Linear Programs

The dual linear program of the primal LP is also given in Figure 2. The key idea is the following: due to weak duality every dual-feasible solution induces an objective value which is at least the number of blocking planes.

As an illustration of the power of our method, let us first reproduce a few known results. The examples should give the reader the correct insight without delving into too much technicality.

Example 3.3 ($c = 0$, the node-blocking case)**.** When $c = 0$, the problem becomes the SNB problem in the node-blocking sense. It has been shown in [21] (which addressed the node-blocking problem in f-cast switches) that $m \geq d^{\lceil (n-1)/2\rceil} + d^{n-\lceil (n-1)/2\rceil} - 1$ is necessary and sufficient for $\log_d(N, 0, m)$ to be SNB.

Let $i_0 = n - \lceil (n-1)/2\rceil$ and $j_0 = \lceil (n-1)/2\rceil$. Assign $u_i = 1$ for all i such that $i_0 \leq i \leq n-1$; and, $v_j = 1$ for all j such that $j_0 \leq j \leq n-1$. All other variables are zeros. It is easy to check that this is a dual-feasible solution with objective value precisely $d^{\lceil (n-1)/2\rceil} + d^{n-\lceil (n-1)/2\rceil} - 2$. Thus, at most 1 more $\text{BY}^{-1}(n)$ is needed, for a total of $d^{\lceil (n-1)/2\rceil} + d^{n-\lceil (n-1)/2\rceil} - 1$ planes in the worst case, matching the known necessary and sufficient condition.

Example 3.4 ($c = n$, the link-blocking case)**.** When $c = n$, the problem becomes the SNB problem in the link-blocking sense. It has been shown in [5, 21] that $m \geq d^{\lceil n/2\rceil -1} + d^{\lfloor n/2\rfloor} - 1$ is necessary and sufficient for $\log_d(N, 0, m)$ to be SNB.

Since there can be no path with $n + 1$ distinct SEs, the variables y_{ij} and z_{ij} are all zeros, so are the l_{ij} and r_{ij}. The primal LP becomes much simpler:

$$\begin{array}{rrl}
\max & \sum_{i+j\geq n} x_{ij} & \\
\text{subject to} & \sum_{j:\, i+j\geq n} x_{ij} \leq d^{n-i} - d^{n-1-i} & \forall i \\
 & \sum_{i:\, i+j\geq n} x_{ij} \leq d^{n-j} - d^{n-1-j} & \forall j \\
 & x_{ij} \geq 0 & \forall i, j
\end{array}$$

The dual LP is:

$$\begin{array}{rcl}
\min & \sum_i (d^{n-i} - d^{n-1-i})u_i + \sum_j (d^{n-j} - d^{n-1-j})v_j & \\
\text{subject to} & u_i + v_j \geq 1 & i + j \geq n \\
 & u_i, v_j \geq 0 & \forall i, j
\end{array}$$

Let $i_0 = \lfloor n/2\rfloor + 1$ and $j_0 = \lceil n/2\rceil$. Assign $u_i = 1$ for all i such that $i_0 \leq i \leq n-1$; and, $v_j = 1$ for all j such that $j_0 \leq j \leq n-1$. All other variables are zeros. This solution is dual-feasible with objective value $d^{\lceil n/2\rceil -1} + d^{\lfloor n/2\rfloor} - 2$. Hence, $m \geq d^{\lceil n/2\rceil -1} + d^{\lfloor n/2\rfloor} - 1$ is sufficient for $\log_d(N, 0, m)$ to be SNB in this case. Again, the sufficient condition obtained with our method matches the known necessary and sufficient condition.

In light of the above examples, we consider $1 \leq c \leq n - 1$ in the rest of this paper.

3.2 The Case When n is Odd

Lemma 3.5. *Suppose $n = 2k + 1$ and $1 \leq c \leq 2k$. For any integer p where $0 \leq p \leq k$, there exists a feasible solution to the dual LP with objective value*

$$2d^k - 2 + \frac{2d^k(d-1)}{c+2} \cdot f(c, d, p),$$

where

$$f(c,d,p) = \sum_{i=0}^{p} \left(\frac{d}{c+1}\right)^i - \frac{1}{d}\sum_{i=0}^{p-1} \frac{1}{[d(c+1)]^i} = \begin{cases} p + 1 - \frac{1-d^{-2p}}{d-\frac{1}{d}} & c+1 = d \\ \frac{1-\left(\frac{d}{c+1}\right)^{p+1}}{1-\frac{d}{c+1}} - \frac{1-\left(\frac{1}{d(c+1)}\right)^p}{d-\frac{1}{c+1}} & c+1 \neq d. \end{cases} \quad (20)$$

Proof. Consider the following assignment to the dual variables:

$$\begin{aligned}
u_i = v_i &= \begin{cases} 1 & k+p+1 \leq i \leq n-1 \\ 1 - \frac{1}{(c+2)(c+1)^{i-k-1}} & k+1 \leq i \leq k+p \\ \frac{1}{(c+2)(c+1)^{k-i}} & k-p \leq i \leq k \\ 0 & 0 \leq i < k-p \end{cases} \\
w_{ij} &= \min\{u_i, u_j\}\ \forall i+j = n-1 \\
s_{ij} = t_{ij} &= 0\ \forall i+j < n-1.
\end{aligned}$$

First, we verify that this assignment is indeed a dual-feasible solution. Note that for $i \leq j$ we have $u_i = v_i \leq u_j = v_j$.

Consider the dual constraint (14). When $i + j \geq n = 2k + 1$, either $i \geq k + 1$ or $j \geq k + 1$. Due to symmetry, we only need to consider $i \geq k+1$. Note that $v_j \geq v_{n-i}$ because $j \geq n-i$. If $i \geq k+1+p$, then $u_i \geq 1$ and thus $u_i + v_j \geq 1$. If $k+1 \leq i \leq k+p$, then $k - p \leq n - i \leq k$. Thus

$$u_i + v_j \geq u_i + v_{n-i} = \left(1 - \frac{1}{(c+2)(c+1)^{i-k-1}}\right) + \frac{1}{(c+2)(c+1)^{k-(n-i)}} = 1.$$

Consider the dual constraint (15). When $i + j = n - 1 = 2k$, either $i \geq k$ or $j \geq k$. Due to symmetry we only need to consider $i \geq k$. If $i \geq k+p+1$, then $j = 2k - i \leq k - p - 1$. Hence, $u_i = 1, v_j = 0, w_{ij} = 0$ and thus the constraint is satisfied. When $k+1 \leq i \leq k+p$, we have $k - p \leq j \leq k-1$. Thus,

$$u_i + v_j + cw_{ij} = \left(1 - \frac{1}{(c+2)(c+1)^{i-k-1}}\right) + (c+1)\frac{1}{(c+2)(c+1)^{k-j}} = 1.$$

When $i = j = k$, we have $u_i = v_j = w_{ij} = 1/(c+2)$. The constraint is straightforwardly verified.

Constraint (16) is verified similarly. Constraint (17) follows from the fact that $u_i \geq w_{i,n-1-i}$ (and all other variables are non-negative). Similarly, constraint (18) follows from the fact that $v_j \geq w_{n-1-j,j}$. Second, it is routine to verify that the objective value of the dual-feasible solution above is precisely $2d^k - 2 + \frac{2d^k(d-1)}{c+2} \cdot f(c,d,p)$. □

Lemma 3.6. *Suppose $n = 2k+1$, $k \geq 2$, and $c = k$. There exists a feasible solution to the dual LP with objective value*

$$2d^k - 2 + \frac{2(d-1)d^{k-2}}{c+2}\left(d^2 - d + \frac{d^3-1}{c+2}\right)$$

Proof. The following solution is dual-feasible with the desired objective value: $u_i = v_i = 1$ for $k+3 \leq i \leq n-1$, $u_{k+2} = v_{k+2} = 1 - \frac{1}{(c+2)^2}$, $u_{k+1} = v_{k+1} = 1 - \frac{1}{(c+2)}$, $u_k = v_k = \frac{1}{(c+2)}$, $u_{k-1} = v_{k-1} = \frac{1}{(c+2)^2}$, $u_i = v_i = 0$ for $0 \leq i \leq k-2$, $w_{kk} = \frac{1}{c+2}$, $w_{k+1,k-1} = w_{k-1,k+1} = \frac{2}{(c+2)^2}$, $w_{k+2,k-2} = w_{k-2,k+2} = \frac{1}{(c+2)^2}$, $w_{ij} = 0$ for all other i,j, $s_{i,k-2} = t_{k-2,i} = \frac{1}{(c+2)^2}$ for $0 \leq i \leq k-2$, $s_{i,k-1} = t_{k-1,i} = \frac{1}{(c+2)^2}$ for $0 \leq i \leq k-2$, and $s_{ij} = t_{ij} = 0$ for all other i,j. □

Lemma 3.7. *Suppose $n = 2k+1$ and $k+1 \leq c \leq 2k$. There exists a feasible solution to the dual LP with objective value*

$$2d^k - 2 + \frac{2(d-1)^2 d^{k-1}}{c+2}.$$

Proof. The following solution is dual-feasible with the desired objective value: $u_i = v_i = 1$ for $k+2 \leq i \leq n-1$, $u_{k+1} = v_{k+1} = 1 - \frac{1}{(c+2)}$, $u_k = v_k = \frac{1}{(c+2)}$, $u_i = v_i = 0$ for $0 \leq i \leq k-1$, $w_{kk} = \frac{1}{c+2}$, $w_{k+1,k-1} = w_{k-1,k+1} = \frac{1}{(c+2)}$, $w_{ij} = 0$ for all other i,j, $s_{i,k-1} = t_{k-1,i} = \frac{1}{(c+2)}$ for $0 \leq i \leq k-1$, and $s_{ij} = t_{ij} = 0$ for all other i,j. □

We are now ready to prove the sufficiency conditions for the odd-n case.

Theorem 3.8. *Consider the case when $n = 2k+1$. Recall that $f(c,d,p)$ is defined in* (20).

(a) If $1 \leq c \leq k-1$, then

$$m \geq 2d^k - 1 + \left\lfloor \frac{2d^k(d-1)}{c+2} \cdot f(c,d,\lfloor \log_d (c+1)/2 \rfloor) \right\rfloor$$

is sufficient for $\log_d(N,0,m)$ *to be c-SNB. In particular, when* $c \le \min\{k-1, d^2-1\}$

$$m \ge 2d^k - 1 + \left\lfloor \frac{2d^k(d-1)}{c+2} \right\rfloor$$

is sufficient.

(b) If $c = k$*, then*

$$m \ge 2d^k - 1 + \left\lfloor \frac{2(d-1)d^{k-2}}{c+2} \left(d^2 - d + \frac{d^3-1}{c+2} \right) \right\rfloor$$

is sufficient for $\log_d(N,0,m)$ *to be c-SNB.*

(c) If $k+1 \le c \le 2k$*, then*

$$m \ge 2d^k - 1 + \left\lfloor \frac{2(d-1)^2 d^{k-1}}{c+2} \right\rfloor$$

is sufficient for $\log_d(N,0,m)$ *to be c-SNB.*

Proof. By Theorem 3.1, the objective value of any dual-feasible solution is an upper bound on the number of $\mathrm{BY}^{-1}(n)$-planes blocking a new request. Hence, one plane more than the objective value is sufficient for the network to be c-SNB.

To see (a), we apply Lemma 3.5. The best dual-feasible solution is the one whose objective value is minimized. In this case, $f(d-1,d,p)$ is minimized at $p_0 = \lfloor \log_d(c+1)/2 \rfloor$. (To see this, notice that $f(c,d,p) \ge f(c,d,p+1)$ iff $p \le \frac{1}{2}\log_d(c+1) - 1$.) Thus, we make use of the dual-feasible solution in Lemma 3.5 corresponding to $p = p_0$. In particular, when $c \le d^2-1$ the function's minimum value is $f(c,d,\lfloor \log_d(c+1)/2 \rfloor) = 1$.

Similarly, (b) follows from Lemma 3.6, and (c) from Lemma 3.7. □

Remark 3.9. Compared to the results of [20] (which was only for the binary network case, i.e. $d = 2$), our sufficient conditions are better when $3 \le c \le k-1$, and are at least as good for other ranges of c.

3.3 The Case When n is Even

Lemma 3.10. *Suppose* $n = 2k$ *and* $1 \le c \le d-1$*. There exists a feasible solution to the dual LP with objective value* $2d^k - 2$*.*

Proof. The following solution is dual-feasible with the desired objective value: $u_i = v_i = 1$ for all $i \ge k$, all other variables are set to 0. □

Lemma 3.11. *Suppose* $n = 2k$ *and* $d \le c \le n-1$*. For any integer* $p, 1 \le p \le k$*, there exists a feasible solution to the dual LP with objective value*

$$d^k + d^{k-1} - 2 + \frac{(d-1)d^{k-1}}{c+2} \cdot (d + g(c,d,p)),$$

where

$$g(c,d,p) = \sum_{i=0}^{p} \left(\frac{d}{c+1}\right)^i - \sum_{i=0}^{p-1} \frac{1}{[d(c+1)]^i} = \begin{cases} p + 1 - \frac{1-d^{-2p}}{1-d^{-2}} & c+1 = d \\ \frac{1-\left(\frac{d}{c+1}\right)^{p+1}}{1-\frac{d}{c+1}} - \frac{1-\left(\frac{1}{d(c+1)}\right)^p}{1-\frac{1}{d(c+1)}} & c+1 \ne d. \end{cases} \tag{21}$$

Proof. The following solution is dual-feasible with the desired objective value: $u_i = 1$ for $k+1 \le i \le n-1$, $u_i = \frac{1}{(c+2)(c+1)^{k-i}}$ for $k-p \le i \le k$, $u_i = 0$ for all other i, $v_j = 1$ for $k+p \le j \le n-1$, $v_j = 1-u_{n-j} = 1-u_{2k-j}$ for $k \le j \le k+p-1$, $v_{k-1} = \frac{1}{c+2}$, $v_j = 0$ for all other j, $w_{ij} = \min\{u_i, u_j\}$ when $i+j = n-1$, and $s_{ij} = t_{ij} = 0$ for all $i+j < n-1$. □

Remark 3.12. The previous two Lemmas hold for any c between 1 and $n-1$, but we stated them in the range of c where they are meant to be applied.

Lemma 3.13. *Suppose $n = 2k$ and $k < c$. There exists a feasible solution to the dual LP with objective value $d^k + d^{k-1} - 2$.*

Proof. The following solution is dual-feasible with the desired objective value: $u_i = v_i = 1$ for $k+1 \le i \le n-1$, $u_k = v_k = \frac{1}{2}$, $u_i = v_i = 0$ for $0 \le i \le k-1$, $w_{k,k-1} = w_{k-1,k} = \frac{1}{2}$, $w_{ij} = 0$ for all other i, j, $s_{i,k-1} = t_{k-1,i} = \frac{1}{2}$ for $0 \le i \le k-1$, and $s_{ij} = t_{ij} = 0$ for all other i, j. □

With the help of Lemmas 3.10, 3.11 and 3.13 the following theorem straightforwardly follows.

Theorem 3.14. *Consider the case when $n = 2k$. Recall that $g(c, d, p)$ is defined in* (21).

(a) If $1 \le c \le d-1$, then $m \ge 2d^k - 1$ is sufficient for $\log_d(N, 0, m)$ to be c-SNB.

(b) If $d \le c \le k$, then

$$m \ge d^k + d^{k-1} - 1 + \left\lfloor \frac{(d-1)d^{k-1}}{c+2} \cdot \left(d + g\left(c, d, \left\lfloor \frac{\log_d(c+1)+1}{2} \right\rfloor \right) \right) \right\rfloor$$

is sufficient for $\log_d(N, 0, m)$ to be c-SNB. In particular, when $d \le c \le \min\{k, d^3 - 2\}$

$$m \ge d^k + d^{k-1} - 1 + \left\lfloor \frac{(d-1)d^k}{c+1} \right\rfloor$$

is sufficient.

(c) If $k < c \le 2k$, then $m \ge d^k + d^{k-1} - 1$ is sufficient for $\log_d(N, 0, m)$ to be c-SNB.

Remark 3.15. Compared to the results of [20] (which was only for the binary network case, i.e. $d = 2$), our sufficient conditions are better when $7 \le c \le k$, and are at least as good for other ranges of c.

4 Necessary Conditions

We have seen in Examples 3.3 and 3.4 that our method gives sufficient conditions which are also necessary when $c = 0$ (node-blocking case) and $c = n$ (link-blocking case). A natural question is: "how good are our sufficient conditions in Theorems 3.8 and 3.14 when $1 \le c \le n-1$?" In this section, we will show that our sufficient conditions are also necessary for $1 \le c \le \min\{k-1, d^2-1\}$ when n is odd, and necessary for $1 \le c \le \min\{k-1, d^3-2\}$ and when $c > n/2$ when n is even.. This is the first time in the literature that any necessary and sufficient conditions are derived for some values of c in the range $1 \le c \ne n$.

Theorem 4.1. *(a) When $n = 2k+1$ and $1 \le c \le \min\{k-1, d^2-1\}$, the sufficient condition*

$$m \ge 2d^k - 1 + \left\lfloor \frac{2d^k(d-1)}{c+2} \right\rfloor$$

of Theorem 3.8 is also necessary for $\log_d(N, 0, m)$ to be c-SNB.

(b) *When $n = 2k$ and $1 \leq c \leq d-1$, the sufficient condition $m \geq 2d^k - 1$ of Theorem 3.8 is also necessary for $\log_d(N,0,m)$ to be c-SNB.*

(c) *When $n = 2k$ and $d \leq c \leq \min\{k-1, d^3-2\}$ the sufficient condition*

$$m \geq d^k + d^{k-1} - 1 + \left\lfloor \frac{(d-1)d^k}{c+1} \right\rfloor$$

of Theorem 3.14 is also necessary for $\log_d(N,0,m)$ to be c-SNB.

(d) *Moreover, when $n = 2k$ and $c \geq k$ the sufficient condition*

$$m \geq d^k + d^{k-1} - 1$$

of Theorem 3.14 is also necessary for $\log_d(N,0,m)$ to be c-SNB.

Proof. For necessity, the main idea is to explicitly construct a network state and a new request for which the number of blocking planes (plus 1) matches the sufficiency conditions. In this sense, the proofs of all parts of this theorem are similar, yet tedious. For the sake of clarity and the fact that the proofs are not very enlightening from a mathematical point of view, we will only provide here a proof of part (a) of this theorem when $d = 2$ and $c = 2$. Hopefully the reader will be able to see the main line of thought. When $d = 2$ and $c = 2$, the sufficient condition becomes $m \geq 2^{k+1} + 2^{k-1} - 1$. The main idea is to create a request $(\mathbf{a}, \mathbf{b})$ and a network state compatible with this request in which there are $2^{k+1} + 2^{k-1} - 2$ blocking planes.

Let $\mathbf{a} = 0^n$ and $\mathbf{b} = 0^n$. Recall our convention that 0^n means a string of n zeros. The blocking planes are constructed as follows.

- For each i where $k+2 \leq i \leq 2k = n-1$, each string $\mathbf{s} \in \mathbb{Z}_2^{2k-i-1}$ and each bit $b \in \mathbb{Z}_2$ create the following requests

$$\begin{array}{rcl} (\mathbf{s}10^i b & , & 0^{2k+1-i}1\mathbf{s}1^{2i-2k-1}b) \\ (1^{2i-2k-1}\mathbf{s}10^{2k+1-i}b & , & 0^i 1\mathbf{s}b) \end{array}$$

  and let each of them be routed through a separate $\text{BY}^{-1}(n)$ plane. The intuition is that, the first request is in $A_i \times B_{n-i}$ and the second is in $A_{n-i} \times B_i$. Hence, by Proposition 2.2 each of the above requests link-blocks $(\mathbf{a}, \mathbf{b})$, all the above planes are blocking planes. The number of blocking planes is

$$2\left(\sum_{i=k+2}^{2k} 2^{2k-i}\right) = 2(2^{k-1} - 1) = 2^k - 2.$$

- Next, we will construct some more blocking planes which route node-blocking requests of type 1. We certainly cannot use any of the inputs and outputs which have already been used to create the blocking planes above. For each string $\mathbf{s} \in \mathbb{Z}_2^{k-2}$ and each symbol $b \in \mathbb{Z}_2$, route the following three requests through a separate plane:

$$\begin{array}{rcl} (\mathbf{s}10^{k+1}b & , & 0^{k-1}1\mathbf{s}00b) \\ (0^k\mathbf{s}10b & , & 0^{k-1}1\mathbf{s}01b) \\ (0^{k-1}\mathbf{s}100b & , & 0^{k-1}1\mathbf{s}10b) \end{array}$$

  It is not difficult to check that each of the above planes are blocking-planes. Basically, the first request is in $A_{k-1} \times B_{k+1}$ which is meant to be a node-blocking request of type 1, the other two requests are both right secondary requests.

Similarly, for each string $\mathbf{s} \in \mathbb{Z}_2^{k-2}$ and each symbol $b \in \mathbb{Z}_2$, route the following three requests through a separate plane:

$$\begin{array}{rcl} (00\mathbf{s}10^{k-1}b, & , & 0^{k+1}1\mathbf{s}b) \\ (10\mathbf{s}10^{k-1}b & , & 01\mathbf{s}0^{k}b) \\ (01\mathbf{s}10^{k-1}b & , & 001\mathbf{s}0^{k-1}b) \end{array}$$

The total number of blocking planes created this way is $2 \cdot 2^{k-1} = 2^k$.

- Finally, for each string $\mathbf{s} \in \mathbb{Z}_2^{k-1}$, route the following three requests through a separate plane:

$$\begin{array}{rcl} (\mathbf{s}10^{k}0 & , & 0^{k}1\mathbf{s}0) \\ (\mathbf{s}10^{k}1 & , & 10^{k-1}1\mathbf{s}0) \\ (\mathbf{s}10^{k-1}10 & , & 0^{k}1\mathbf{s}1) \end{array}$$

  It is not difficult to check that each of the above planes are blocking-planes. Basically, the first request is in $A_k \times B_k$ which is meant to be a node-blocking request of type 1, the other two requests are left and right secondary requests. The total number of blocking planes created this way is 2^{k-1}.

We have not used any input nor output twice in creating a total of $2^k - 2 + 2^k + 2^{k-1} = 2^{k+1} + 2^{k-1} - 2$ blocking planes. The necessary condition is thus established. $\square$

References

[1] Chen, H.-B., and Hwang, F. K. On multicast rearrangeable 3-stage clos networks without first-stage fan-out. *SIAM Journal on Discrete Mathematics 20*, 2 (2006), 287–290.

[2] Chinni, V., Huang, T., Wai, P.-K., Menyuk, C., and Simonis, G. Crosstalk in a lossy directional coupler switch. *J. Lightwave Technol. 13*, 7 (1995), 1530–1535.

[3] D.Li. Elimination of crosstalk in directional coupler switches. *Optical Quantum Electron., 25*, 4 (1993), 255–260.

[4] F.K.Hwang, and Lin, B.-C. Wide-sense nonblocking multicast $\log_2(n, m, p)$ networks. *IEEE Transactions on Communications 51*, 10 (Oct 2003), 1730–1735.

[5] F.K.Hwang, Wang, Y., and Tan, J. Strictly nonblocking f-cast $\log_d(n, m, p)$ networks. *IEEE Transactions on Communications 55*, 5 (May 2007), 981–986.

[6] Hwang, F. Choosing the best $\log_2(n, m, p)$ strictly nonblocking networks. *IEEE Transactions on Communications 46*, 12 (Dec 1998), 454–455.

[7] Hwang, F. K. *The mathematical theory of nonblocking switching networks*. World Scientific Publishing Co. Inc., River Edge, NJ, 2004.

[8] Jiang, X., Pattavina, A., and S.Horiguchi. Rearrangeable f-cast multi-$\log 2n$ networks. *IEEE Transactions on Communications* (2007). to appear.

[9] Jiang, X., Shen, H., ur Rashid Khandker, M. M., and Horiguchi, S. Blocking behaviors of crosstalk-free optical banyan networks on vertical stacking. *IEEE/ACM Transactions on Networking 11*, 6 (2003), 982–993.

[10] KABACINSKI, W., AND DANILEWICZ, G. Wide-sense and strict-sense nonblocking operation of multicast multi-$log_2 n$ switching networks. *IEEE Transactions on Communications 50*, 6 (Jun 2002), 1025–1036.

[11] LEA, C.-T. Muti-$\log_2 n$ networks and their applications in high speed electronic and photonic switching systems. *IEEE Transactions on Communications 38*, 10 (1990), 1740–1749.

[12] LEA, C.-T., AND SHYY, D.-J. Tradeoff of horizontal decomposition versus vertical stacking in rearrangeable nonblocking networks. *IEEE Transactions on Communications 39* (1991), 899–904.

[13] MAIER, G., AND PATTAVINA, A. Design of photonic rearrangeable networks with zero first-order switching-element-crosstalk. *IEEE Trans. Comm. 49*, 7 (Jul 2001), 1248–1279.

[14] MUKHERJEE, B. *Optical Communication Networks*. McGraw-Hill, New York, NY, 1997.

[15] NGO, H. Q., AND DU, D.-Z. Notes on the complexity of switching networks. In *Advances in Switching Networks*, D.-Z. Du and H. Q. Ngo, Eds., vol. 5 of *Network Theory and Applications*. Kluwer Academic Publishers, 2001, pp. 307–367.

[16] PATTAVINA, A., AND TESEI, G. Non-blocking conditions of multicast three-stage interconnection networks. *IEEE Transactions on Communications 46*, 4 (Dec 2005), 163–170.

[17] SHYY, D.-J., AND LEA, C.-T. $\log_2(n, m, p)$ strictly nonblocking networks. *IEEE Transactions on Communications 39*, 10 (1991), 1502–1510.

[18] STERN, T. E., AND BALA, K. *Multiwavelength Optical Networks: A Layered Approach*. Prentice Hall PTR, Upper Saddle River, NJ, 1999.

[19] VAEZ, M., AND LEA, C.-T. Wide-sense nonblocking Banyan-type switching systems based on directional couplers. *IEEE J. Select. Areas Commun. 16*, 7 (Sep 1998), 1327–1332.

[20] VAEZ, M. M., AND LEA, C.-T. Strictly nonblocking directional-coupler-based switching networks under crosstalk constraint. *IEEE Trans. Comm. 48*, 2 (Feb 2000), 316–323.

[21] WANG, Y., NGO, H. Q., AND JIANG, X. Strictly nonblocking f-cast d-ary multilog networks under fanout and crosstalk constraints. In *Proceedings of the 2008 International Conference on Communications (ICC)* (Bejing, China, 2008), IEEE.

[22] WU, J.-C., AND TSAI, T.-L. Low complexity design of a wavelength-selective switch using raman amplifiers and directional couplers. In *GLOBECOM* (2006).