On Noninterruptive Rearrangeable Networks

Frank K. Hwang, Wen-Dar Lin, and Vadim Lioubimov

***Abstract*—In this paper, we study a new class of nonblocking networks called noninterruptive rearrangeable (NIR) networks, which are rearrangeable under the additional condition that existing connections are not interrupted while their paths being possibly rerouted to accommodate a new request. We give a complete characterization of NIR Clos networks built of switching elements of various nonblocking properties. In particular, we propose a novel class of NIR Clos networks that leads to recursive constructions of various cost-efficient multistage NIR networks. Finally, we present examples of such constructions and compare them with the best previously known results.**

***Index Terms*—Clos network, doubled path, noninterruptive rearrangeable (NIR), output (input)-divertability, Paull's matrix, rearrangeably nonblocking (RNB), strictly nonblocking (SNB), wide-sense nonblocking (WSNB).**

I. Introduction

A *request* is a pair of input and output requesting a connection; once *routed*, a request turns into a *connection*, which is a path link-disjoint to all other connections. Traditionally, we classify nonblocking interconnection networks into three levels, that is: *strictly nonblocking (SNB)*, *wide-sense nonblocking (WSNB)*, and *rearrangeably nonblocking (RNB)* or *rearrangeable*. A network is SNB if a request can always be routed by a path link-disjoint to all existing connections, regardless of how they are routed. A network is WSNB if the above can be achieved under a routing algorithm. A network is RNB if link-disjoint paths exist for any set of requests (routed simultaneously). Another equivalent definition of the RNB network is: a request can always be routed under the condition that existing connections are allowed to be *rearranged* (rerouted). Currently, not much is known about WSNB networks. Generally speaking, an SNB network involves more hardware (cost), sometimes more than twice than an RNB network, and an RNB network may involve some delay before a given request is routed where an SNB network can always route a request immediately.

Besides the delay problem, an RNB network usually needs to disconnect and to reroute some existing connections before routing a request (see Fig. 1). In many applications, the main concern in using RNB networks is that the disconnected connections may be lost during the interruption period. In this paper, we study a new class of nonblocking networks called *noninterruptive rearrangeable (NIR)* networks, which are rearrangeable under the additional condition that no connection is taken down before its rerouted path has already been connected (thus, a *doubled path* is formed momentarily, although the new path can share some internal links with the original path). This implies that the rearrangement operation is done sequentially along with incoming requests, not the case that all requests are present simultaneously. Fig. 2 illustrates an example for such a process.

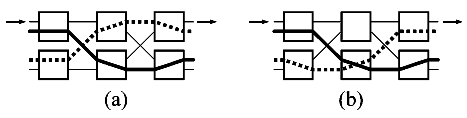
Fig. 1. Well-known RNB network, the Benes network of size 4×4. (a) The indicated request cannot be routed directly. (b) The request can be routed after the dot-line connection is rearranged (disconnected-and-rerouted).

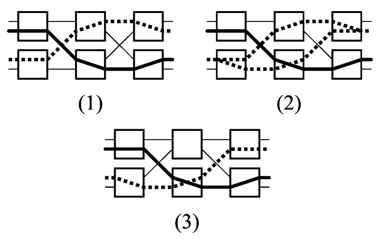
Fig. 2. Rearrangement with a doubled path.

In 1994, Bowdon [2] first proposed a three-stage Clos NIR network. However, the network requires all its switching elements (switches) to have the NIR property when recursively constructing NIR Clos networks for more than three stages. In this paper, we give a complete characterization of three-stage NIR Clos networks based on switches of various nonblocking properties. In particular, we propose a novel network in this class without requirement on the middle switches to be NIR. We show that this network allows recursive construction of multistage NIR networks that are more cost-efficient than the ones based on Bowdon's network.

The remainder of this paper is organized as follows. In Section II, we introduce some background knowledge. In Sections III and IV, we introduce and characterize various classes of three-stage NIR Clos networks. In Section V, we discuss an additional nonblocking property of the introduced NIR Clos networks that is needed for further constructions of multistage NIR networks. In Section VI, we present examples of such constructions and compare them with the best previously known results.

II. Terminology and Preliminary

The three-stage *Clos network* $C(n,m,r)$ is one of the most basic multistage interconnection networks. The first stage of $C(n,m,r)$ consists of r $n \times m$ switches, the second stage m $r \times r$ switches, the third stage r $m \times n$ switches, and the linking

Manuscript received April 19, 2003; revised June 23, 2005; approved by IEEE/ACM Transactions on Networking Editor F. Neri.

F. K. Hwang is with the Department of Applied Mathematics, National Chiao Tung University, HsinChu 30050, Taiwan, R.O.C. (e-mail: fhwang@math.nctu.edu.tw).

W.-D. Lin is with the Institute of Information Science, Academia Sinica, Taipei NanKang 115, Taiwan, R.O.C. (e-mail: wdlin@iis.sinica.edu.tw).

V. Lioubimov is with Almipa Inc., Concord, ON, L4K 1W3 Canada (e-mail: v.lioubimov@sympatico.ca).

Digital Object Identifier 10.1109/TNET.2006.882846

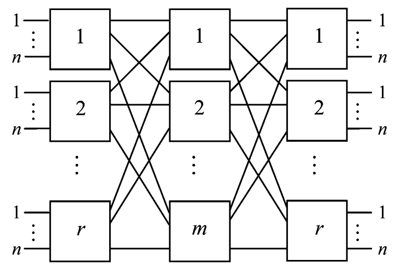
Fig. 3. Three-stage Clos network $C(n, m, r)$. The rectangles are switches.

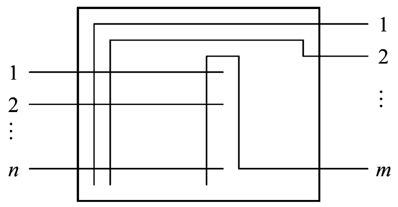
Fig. 4. The crossbar $X_{n,m}$ that can arbitrarily establish connections from the n inlets to the m outlets. The crossbar is one of the most basic switches.

patterns of adjacent stages are complete bipartite graphs (see Fig. 3). Note that a switch may be a crossbar (see Fig. 4) or a network.

Clearly, for any two connections in a Clos network passing the same switch in the first stage (*input switch*), they must go to different switches in the second stage (*middle switches*); the same is true for every switch in the third stage (*output switch*). We can use *Paull's matrix* [4], which is an $r \times r$ matrix, to record the *network state* of a Clos network $C(n, m, r)$: cell (i, j) records the middle switches used by connections passing the ith input switch and the jth output switch. Traditionally, letters (i.e., a, b, $c\ldots$) but not numbers are used to represent the middle switches. Due to the fact that every input (output) switch has exactly n inlets (outlets), the Paull's matrix must have the following properties.

1) At most m distinct letters are used in the whole matrix.
2) Every letter appears at most once in each row (column).
3) Each row (column) contains at most n letters.

Fig. 5(a) illustrates an example of a Clos network $C(3, 3, 4)$ and its corresponding Paull's matrix. Note that a letter, say c, in cell (i, j) represents a connection from input switch i, through middle switch c, to output switch j. The following theorem is well known [3].

Theorem 1: Suppose that all switches of $C(n, m, r)$ are RNB. Then, the network $C(n, m, r)$ is RNB if and only if $m \geq n$.

An important issue for RNB networks is to reduce the number of rearrangements when routing a request. We introduce some previous results on RNB networks for the self-containment of this paper. For a Clos network $C(n, m, r)$, observe that, when there is a request from input switch i to output switch j, there must be at most $n - 1$ letters in row i of the Paull's matrix, and the same is true for column j. There are two cases to be considered: either R1) there exists some letter, say c, not in row i nor column j, or R2) there is no such letter as described in R1. For case R1, we can route the request through the middle switch c. For case R2, there exist some two letters, say a and b, such that a appears in row i but not column j, b in column j but not row i. Paull's method [4] is to rearrange the connection represented by the a in row i to be carried by b, then the request can be routed by a. However, there may be another b in the same column as the first a, which means that we must rearrange the connection represented by the second b to be carried by a. Again, there may be another a in the same row as the second b, then we have to rearrange the connection represented by the second a to be carried by b, and so on. If we join every two consecutive letters in the above process by an arc from the previous one to the latter, then we obtain a path, called the *ab-path*, which alternates in vertical and horizontal turns (with alternating a's and b's; see Fig. 5(b) for an example). Observe that such a path never forms a loop and the total number of a's and b's in a Paull's matrix are limited to $2r - 2$, i.e., the number of rearrangements are limited to $2r - 2$. Later, Benes [1] improved it to $r - 1$ based on the observation that the *ab*-path and the *ba*-path are disjoint, so we can choose either one for rearrangement. Fig. 5(c) illustrates the *ab*-path and the *ba*-path if the request is from the first input switch to the first output switch. For convenience, we always assume that a is the letter that appears in row i but not column j and b in column j but not row i for case R2.

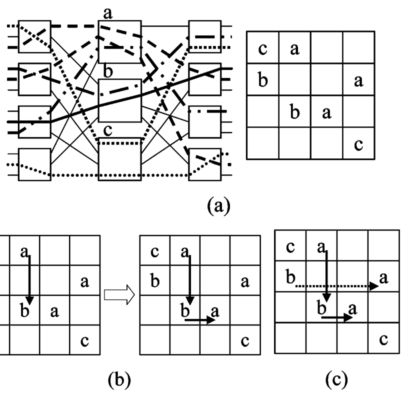

Fig. 5. (a) Example of a Clos network $C(3, 3, 4)$ and its corresponding Paull's matrix. (b) Example of *ab*-path for a request from the first input switch to the first output switch. (c) The *ab*-path (arrows) and the *ba*-path (dot-line arrow) for a request from the first input switch to the first output switch.

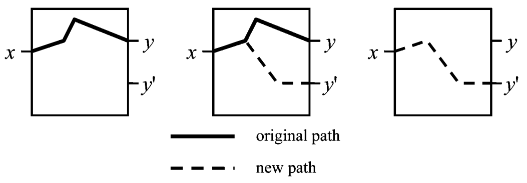

Fig. 6. Output-divertibility.

In order for a network to be NIR, some of its switches must possess the *output (input)-divertability* property. A network is said to be *output (input)-divertible* if we can always noninterruptively append a new path from the input (output) of a given existing connection to some unused output (input) when all existing connections are one-to-one (see Fig. 6). This property is weaker than the two-cast property since the performance is required only when all existing connections are one-to-one, while the two-cast performance is required when existing connections

c	a		
b	c	a	
	b	c	a
		b	c

Fig. 7. Network state providing that $C(3,3,4)$ is RNB, but not NIR.

are all two-cast (every connection has one input and two outputs). It is also slightly different from the noninterruptive rearrangeability as the second path goes to a different (unspecified) output.

As mentioned in Section I, "noninterruptive" means that we must build a doubled path for every existing connection to be rearranged. Also, it should be noticed that it is not always possible to build doubled paths in an RNB network. For example, the Paull's matrix in Fig. 7 represents the network state of a Clos network $C(3,3,4)$ such that we cannot rearrange any existing connection by a doubled path, which means that we cannot connect the request from the first input switch to the first output switch without interruption.

An important characteristic of our constructions is that we maintain a set of *reserved* middle switches in addition to the requirement of RNB network, where the reserved middle switches are empty switches in their normal states. Also, doubled paths are built with the aids of reserved middle switches. For instance, a reserved middle switch is used to simultaneously connect the second paths (perhaps plus the request) of existing connections. This operation is always doable because a reserved middle switch has free links to all input (or output) switches. A reserved switch after being used to route paths is no longer reserved and must be replaced by an empty switch which can be obtained by having all its paths rerouted. Note that the output (input)-divertability is required for input (output) switches if we want to build doubled paths for connections in a three-stage Clos network. We assume that all crossbars are with input/output-divertability.

Lemma 1: Suppose that a three-stage Clos network satisfies the following conditions: 1) all switches are NIR and 2) the input (output) switches are output (input)-divertible. Then, a doubled path can be built for a connection from input switch i to output switch j if and only if there is a letter not appearing in row i nor column j of the Paull's matrix.

Proof: Suppose that an appended path can be built to the specified connection. Note that this path must use a different middle switch, say c, from that of the original path, otherwise the two paths are identical. Then, in order for c to be accessible for both switches i and j, it should carry neither a connection from switch i, nor one to output switch j. This means that the letter c appears neither in row i nor column j of the Paull's matrix. Conversely, suppose there is a letter c that appears neither in row i nor column j. Then, by similar arguments, it is clear that the path from input switch i to output switch j through middle switch c can be an appended path to the specified connection. Q.E.D.

III. The One-Reserved Algorithm

Theorem 2: Suppose that a network $C(n,n,r), n \geq 2, r \geq 2$, satisfies the following conditions: 1) all switches are NIR and 2) the input (output) switches are output (input)-divertible. Then the network $C(n,n,r)$ is NIR if and only if $r = 2$.

Proof: Sufficiency: Consider the 2×2 Paull's matrix representing the corresponding network state. Without loss of generality, we assume that the request is from the first input switch to the first output switch and is blocked, i.e., there is no letter c not in first row nor first column (case R2). Since b appears in cell (2, 1) of the Paull's matrix, then b must not appear in cell (1, 1), nor cell (2, 2). By assumption, a must lie in cell (1, 2) (or we would have case R1) and b must not. By Lemma 1, we can then build a doubled path through b for the connection represented by a in cell (1, 2); this is doable by the NIR assumption for b that it may carry some existing connections. After we delete a from cell (1, 2), the request in (1, 1) can be routed by a.

Necessity: Consider the following Paull's matrix of size $r \times r$, $r \geq 3$, where C denotes all middle switches other than a or b:

$$\begin{array}{c|c|c|c|c}
C & a & & & \\ \hline
b & C & a & & \\ \hline
 & b & C & \ddots & \\ \hline
 & & \ddots & \ddots & a \\ \hline
 & & & b & C
\end{array}$$

Since $r \geq 3$, then it is easy to observe that every letter is *collinear*, i.e., two letters are on the same row or the same column, to any other letter. By Lemma 1, this means no doubled paths can be built in this Paull's matrix, that is, the corresponding $C(n,n,r)$ is not NIR. Q.E.D.

Bowdon proved [2] that $C(n, n+1, r)$ is an NIR network with a routing algorithm that requires at most $1.5r$ rearrangements to connect a new request. Using a different $1.5r$-rearrangement algorithm for $C(n, n+1, r)$, we prove the following theorem.

Theorem 3: Suppose that a network $C(n,m,r), n \geq 2, r \geq 3$, satisfies the following conditions: 1) all switches are NIR and 2) the input (output) switches are output (input)-divertible. Then, the network $C(n,m,r)$ is NIR if and only if $m \geq n+1$.

Proof: Sufficiency: We prove the sufficiency by showing that the following algorithm works correctly for every incoming request.

One-Reserved Algorithm: Suppose that there is one reserved middle switch, say d. Without considering d, the Clos network can be treated as an RNB network since $m \geq n+1$. Using the Paull's matrix, we consider cases R1 and R2.

Case R1: Route the request by c

Case R2: Partition all a's and b's into three parts: 1) of the *ab*-path; 2) of the *ba*-path; and 3) not in the *ab*-path nor the *ba*-path. Do the following steps.

a) Rearrange all b's in {part 1, part 3} and all a's in part 2 to d's by doubled paths and connect the request by d.

b) Rearrange all b's in part 2 to a's by doubled paths.

To show the correctness of the one-reserved algorithm, it is sufficient to prove that: 1) there is always an empty middle switch after connecting a request and 2) all the operations in the one-reserved algorithm are doable. To this end, we use induction.

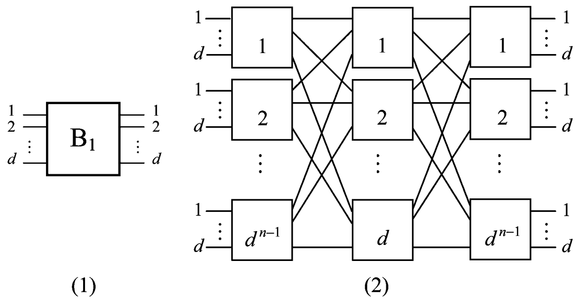

Fig. 8. Recursive construction of d-ary Benes network. (1) B_1, a $d \times d$ crossbar with d^2 crosspoints. (2) B_n, constructed as $C(d, d, d^{n-1})$, where the input, output, and middle switches are B_1, B_1, and B_{d-1}, respectively. B_n has d^n inputs/outputs and $d^n(2n-1)$ crosspoints.

Base case: a network carrying no connection ensures an empty middle switch.

Induction step: as described in the algorithm, suppose there is a reserved middle switch d and treat the network as an RNB network without considering d. Using Paull's matrix, for case R1, we connect the request by c and then d is kept reserved. For case R2, since d does not appear in the Paull's matrix, and all positions of: 1) b's in {part 1, part 3}; 2) a's in part 2; and 3) the requests are mutually noncollinear. By Lemma 1, we can rearrange these connections by doubled paths and connect the request. Now, because all a's in part 2 are rearranged to d's, all b's in part 2 can be rearranged to a's by doubled paths and then b becomes reserved, ensuring the existence of an empty middle switch. One should notice that, by the NIR assumption, the second paths can be appended to a without causing any interruption.

Necessity: By the counterexample in the proof of Theorem 2. Q.E.D.

Note that, instead of rearranging b's in part 3 to d's and b's in part 2 to a's, we can also rearrange a's in part 3 to d's and a's in part 1 to b's. Suppose the lengths of the *ab*-path and the *ba*-path are x and y, respectively. In step a), the number of rearrangements applied to {part 1, part 3} is limited to $\frac{x+y}{2}$. Furthermore, the number of all the other rearrangements are limited to $\frac{2r-2-x-y+((x+y+2)/2)}{2}$ (there are at most $2r-2-x-y$ a's and b's in part 3 and at most $\frac{x+y+2}{2}$ a's and b's in {part 1, part 3} not rearranged by step a). Recalled that $x+y \leq 2r-2$ since the disjointness of the *ab*-path and the *ba*-path, we conclude that the total number of rearrangements are limited to $1.5r$.

The cost (we define the cost of a network as the number of *crosspoints*) of an $N \times N$ crossbar is N^2, which may be too large when N large. Thus, we may want to use some low-cost networks as switches in the Clos network, e.g., the *Benes networks*. Fig. 8 shows the recursive construction of generalized d-ary Benes networks. However, the lower cost usually means a weaker nonblocking property, so we have to characterize the nonblocking property of a Clos network $C(n, m, r)$ composed by switches that are not crossbars.

Theorem 4: $C(1, 2, r)$ is NIR if: 1) the middle switches are RNB and 2) the input (output) switches are output (input)-divertible.

Proof: Use one of the two middle switches as the *active* switch carrying all of the connections and keep the other switch reserved. For any incoming request, route all existing connections and the request in the reserved switch. Then take down the existing connections from the active switch and thus interchange the roles of the active switch and the reserved switch. All of the above operations are doable because the middle switches are RNB and the output (input) switches have the required input (output)-divertability. Q.E.D.

It should be noticed that either all input or all output switches are redundant in the NIR network $C(1, 2, r)$ and can be replaced by the corresponding 1×2 or 2×1 joints, which however may not be feasible for some applications.

IV. The Two-Reserved Algorithm

An NIR middle switch can always carry a new connection rearranged to it, but an RNB middle switch cannot. This is because an RNB network may get into a blocking state as described in Theorem 2 under the noninterruptive condition. Theorems 3 and 5 show the difference between using NIR and RNB networks as middle switches.

Note that, in the one-reserved algorithm, we rearrange some connections carried by b to be carried by a while a is possibly carrying some other connections. This is why the middle switches are required to be NIR. When the middle switches have only the RNB property, we use a reserved middle switch to route the second path of a double path, plus perhaps the request, just like in the algorithm for $C(1, 2, r)$ in the proof of Theorem 4.

Theorem 5: Suppose that a network $C(n, m, r)$, $n \geq 2$, $r \geq 3$, satisfies the following conditions: 1) the middle switches are RNB and 2) the input (output) switches are NIR with output (input)-divertability. Then the network $C(n, m, r)$ is NIR if $m \geq n+2$.

Proof: We prove this theorem by showing that the following algorithm works correctly for every incoming request.

Two-Reserved Algorithm: Suppose there are two reserved middle switches, say d and e. Without considering d and e, the Clos network can be treated as an RNB network since $m \geq n+2$. Using Paull's matrix, now we consider cases R1 and R2.

Case R1: Use the reserved middle switch d to route the request plus all connections in c, and then take down all connections through c.

Case R2: Partition all a's and b's as in the one-reserved algorithm. Then do the following steps:

a) In middle switch d, route the connections carried by a in {part 1, part 3} and by b in part 2, then take down their old paths.
b) In middle switch e, route the request plus connections carried by b in {part 1, part 3} and by a in part 2, then take down the old paths of existing connections.

Since d does not appear in the Paull's matrix and all the positions of a's in {part 1, part 3} and b's in part 2 are mutually noncollinear, by Lemma 1, we can rearrange the above connections by doubled paths to d. Similarly, the request and the connections represented by b's in {part 1, part 3} and a's in part 2 can be carried by e without interrupting. Note that for case R1, c and e are the resulting reserved middle switches; and for case R2, a and b are the resulting reserved middle switches. Q.E.D.

Since we rearrange all the a's and b's in the Paull's matrix exactly once, the number of rearrangements is $2r - 2$ in the worst case.

V. Divertability of NIR Networks

To construct a very large NIR network, we may want to recursively use NIR (or RNB) networks as the input/middle/output switches. However, by Theorems 3–5, input (output) switches of a NIR Clos network are required to have output (input)-divertability, which means that an NIR network may not be suitable as an input (output) switch in a larger network. On the other hand, an NIR network may not be NIR anymore if forced to perform some diverting operations (for example, a reserved middle switch disappears). Fortunately, we will show that all of the NIR networks we introduced before are output (input)-divertible. Without loss of generality, the following theorems consider only the output-divertability.

Theorem 6: $C(1, 2, r)$ is NIR with output-divertability if: 1) the middle switches are RNB and 2) the input (output) switches are output (input)-divertible.

Proof: As in Theorem 4, use the active switch to carry the set C of all existing connections and use the reserved switch to carry $(C \cup (x, y')) \backslash (x, y)$, where (x, y') is the input-output pair of the appended path and y is the output of the original path. Then we can just take down paths using the active switch to let all connections become one-to-one; the reserved switch becomes active and the active switch becomes reserved. Thus, the network is NIR with output-divertability. Q.E.D.

Theorem 7: $C(n, n, 2)$ is NIR with output-divertability if: 1) all switches are NIR, 2) the input switches and middle switches are output-divertable, and 3) the output switches are input/output-divertable.

Proof: We assume that the diverted connection is from the first input switch, through d, to the second output switch, and the new output is in first output switch. Now we ignore the d in cell $(1, 2)$ and then treat this problem as to noninterruptively connect a new request from the first input switch to the first output switch. For case R1, append a new path by going through c, where c and d are not necessarily different. For case R2, rearrange the b to a by a doubled path, and then append a new path by going through b where b and d are not necessarily different. Afterwards, we take down the links belonging to the original path but not on the new path. Then the network is again NIR. Q.E.D.

Theorem 8: $C(n, n + 1, r)$ is NIR with output-divertability if: 1) all switches are NIR; 2) the input switches are output-divertable; and 3) the output switches are input/output-divertable.

Proof: Again, we ignore the letter carrying the original path and then treat this problem as to noninterruptively connect a new request from the original input to the new output. Now we perform the one-reserved algorithm except not to disturb the original path. Since the new path will be carried by a middle switch that carries nothing at the beginning, the middle switches used by the new path and the original path are different, that is, the divertability is not required for middle switches. Afterwards, we take down the original path, then the existence of a reserved middle switch is ensured, implying that the network is again NIR. Q.E.D.

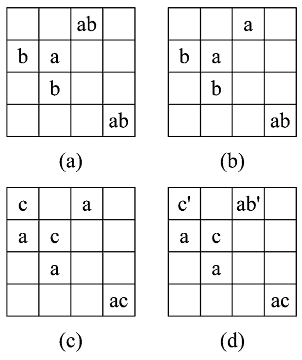

Fig. 9. (a) Paull's matrix at the beginning. Note that the letter c is unused. (b) The letter b in cell (1, 3) is ignored. (c) After performing the one-extra algorithm. (d) The actual network state after performing the diverting algorithm; b' and c' represent the one-to-two connection. Note that b will be free if the original path is taken down.

TABLE I
Required Input/Output-Divertability for Switches of NIR Clos Networks

NIR network	Network divertability	Required divertability for switches		
		Input switches	Middle switches	Output switches
C(1, 2, *r*)	output	output	RNB	input
	input/output	output	RNB	input
	input	output	RNB	input
C(*n*, *n*, 2)	output	output	output	input/output
	input/output	input/output	input/output	input/output
	input	input/output	input	input
C(*n*, *n*+1, *r*)	output	output	NIR	input/output
	input/output	input/output	NIR	input/output
	input	input/output	NIR	input
C(*n*, *n*+2, *r*)	output	output	RNB	input/output
	input/output	input/output	RNB	input/output
	input	input/output	RNB	input

Theorem 9: $C(n, n + 2, r)$ is NIR with output-divertability if: 1) middle switches are RNB; 2) the input switches are NIR with output-divertability; and 3) the output switches are NIR with input/output-divertability.

Proof: By a similar argument as in the proof of Theorem 8. Q.E.D.

Fig. 9 illustrates an example of $C(2, 3, 4)$ for Theorem 8 that the original path is carried by b in cell (1, 3) and the new output is in the first output switch.

Table I gives the required input/output-divertability for switches of Clos networks with different divertabilities. It should be noticed that Theorems 6–9 give only sufficient conditions for divertabilities.

VI. Construction Examples and Cost Comparison

Theorems 2–9 allow flexible recursive construction of various multistage NIR networks, some of which are more cost-efficient than the similar networks based on the previously known results. In this section, we present kinds of constructions under different restrictions on the crossbars sizes.

A. Arbitrary $m \times n$ Crossbars are Allowed

To give a fair comparison, we generate a set of NIR networks according the following rules. From Theorems 1, 3, and 5, we know the following.

1) *(RNB expansion)* $C(N, N, R)$ using RNB $R \times R$ middle switches and crossbars as input/ouput switches is RNB.
2) *(Plus-1 expansion)* $C(N, N+1, R)$ using NIR $R \times R$ middle switches and crossbars as input/ouput switches is NIR.
3) *(Plus-2 expansion)* $C(N, N+2, R)$ using RNB $R \times R$ middle switches and crossbars as input/ouput switches is NIR.

For convenience, we call N the *expanding factor*. By these expansion methods, we could generate all $(2x+1)$-stage NIR Clos networks by using $(2x-1)$-stage networks as middle switches and crossbars as input/output switches, that is, we start with a crossbar of size N_0 and recursively apply networks of size $\prod_{i=o}^{x-1} N_i$ as middle switches for x-th expansion with expansion factor N_x. Thus, we know that every aimed NIR network with $(2x-1)$ stages could be represented by a series of x expansion factors and a series of $x-1$ expansion types. By using the same expansion factor N_i, it should be noticed that Plus-1 expansion will be the only reasonable choice for expanding a NIR network because Plus-2 expansion will result in more cost and RNB expansion will result in a non-NIR network (Theorem 2). Consequently, the second series of $x-1$ expansion types should be in the form of

$$(\ldots,\text{RNB expansion},\text{Plus-2 expansion},\text{Plus-1 expansion},\ldots)$$

where the number of RNB expansion (or Plus-1 expansion) might be 0 and the number of Plus-2 expansion should be 1 or 0, that is, this series could be replaced by a number ranged from 0 to $x-2$ that indicates the last RNB expansion, i.e., 0 means no RNB expansion (and thus no Plus-2 expansion), $x-2$ means no Plus-1 expansion and Plus-2 is the final expansion. By so doing, we are able to use a series of x expansion factors and one additional number to represent an aimed NIR network. For example, (4, 5, 6, 7, 8)-0 means a NIR network constructed by the following steps.

0) A 4×4 crossbar.
1) Respectively use $5 \times (5+1)$, 6×5, 4×4 crossbars as input, output, mid switches to construct a NIR network (a Plus-1 expansion).
2) Respectively use $6 \times (6+1)$ crossbars, 7×6 crossbars, networks by step 1 as input, output, mid switches to construct a NIR network (a Plus-1 expansion).
3) Respectively use $7 \times (7+1)$ crossbars, 8×7 crossbars, networks by step 2 as input, output, mid switches to construct a NIR network (a Plus-1 expansion).
4) Respectively use $8 \times (8+1)$ crossbars, 9×8 crossbars, network in step 3 as input, output, mid switches to construct a NIR network (a Plus-1 expansion).

For another example, (4, 5, 6, 7, 8)-2 means an NIR network constructed by the following steps.

0) A 4×4 crossbar.
1) Respectively use 5×5, 5×5, 4×4 crossbars as input, output, mid switches to construct an RNB network (an RNB expansion).
2) Respectively use 6×6 crossbars, 6×6 crossbars, networks by step 1 as input, output, mid switches to construct an RNB network (an RNB expansion).
3) Respectively use $7 \times (7+2)$ crossbars, 9×7 crossbars, networks by step 2 as input, output, mid switches to construct a NIR network (a Plus-2 expansion).
4) Respectively use $8 \times (8+1)$ crossbars, 9×8 crossbars, network in step 3 as input, output, and mid switches to construct a NIR network (a Plus-1 expansion).

To get an overview on the efficiency of NIR networks, we generate all networks using above expansion methods by exhaustively generating all expansion factor series with products $4, 8, 16, \ldots, 2^{15} = 32768$ and assigning the additional numbers according the sizes of corresponding series. For convenience, we called a generated network as a *Plus-1 network* if its last expansion is Plus-1 expansion, and *Plus-2 network* if its last expansion is Plus-2 expansion. We also apply a similar technique to generate RNB networks of sizes $4, 8, 16, \ldots, 32768$ by using only RNB expansion, then construct NIR networks by Theorem 4 and classify them as *2-RNB networks* (using two copies of RNB networks in middle).

In Fig. 10, we give the size-(cost/size) plot of best selections of various sizes and different last applied constructions. It could be observed that Plus-2 networks are the most cost-efficient NIR networks for size greater than 256. Table II gives a detailed comparison for Plus-1 and Plus-2 networks larger than 256. Notice that larger Plus-1 networks usually need Plus-2 expansion to gain better efficiency. Also notice that a potential advantage of Plus-2 expansion is to use the most compact RNB networks like Waksman networks [5] as middle switches. Fig. 11 gives a general construction of Waksman networks, and the black boxes in Fig. 10 are *Plus-2* networks applying Waksman's construction.

B. Only 1×2, 2×2, and 2×1 Crossbars Are Allowed

In this subsection, we give an algorithm to generate NIR networks under the restriction that only 1×2, 2×2 and 2×1 crossbars are allowed. This algorithm is based on the following facts.

1) 2-$B_n(C(1, 2, 2^n)$ using binary Benes network B_n as middle switches) is NIR with input/output-divertability (by Theorems 4 and 6).
2) NIB_n (described in Fig. 12) is NIR with input/output-divertability (by Theorem 2 and 7).
3) $C(n, n+1, r)$ is NIR with input/output-divertability if the input(output) switches are NIR with input/output-divertability and the middle switches are NIR (by Theorem 3 and 8).

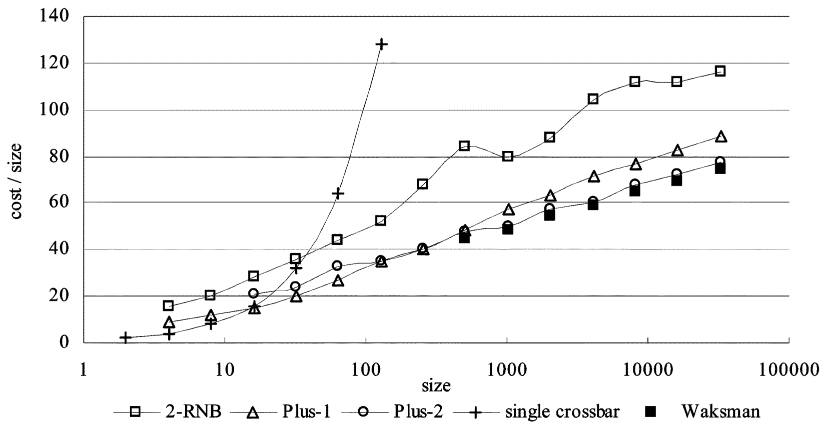

Fig. 10. Best networks of different sizes and different constructions. Each point is marked by its last applied expansion. Black boxes are Plus-2 networks applying Waksman's construction.

TABLE II
DETAILED COMPARISON BETWEEN PLUS-1 AND PLUS-2 NETWORKS

Network size	Plus-1 networks			Plus-2 networks			Plus-2 versus Plus-1
	cost	stages	construction	cost	stages	construction	
512	24768	5	(3, 3, 3)-0	24320	9	(1, 1, 2, 2, 3)-3	0.98
1024	58608	7	(2, 2, 3, 3)-0	51200	9	(2, 2, 1, 2, 3)-3	0.87
2048	129024	7	(3, 2, 3, 3)-1	117760	11	(1, 1, 2, 2, 2, 3)-4	0.91
4096	292608	11	(1, 1, 2, 2, 3, 3)-3	245760	11	(2, 1, 2, 2, 2, 3)-4	0.84
8192	631296	11	(1, 2, 2, 2, 3, 3)-3	552960	13	(1, 1, 2, 2, 2, 2, 3)-5	0.88
16384	1354752	13	(1, 1, 2, 2, 2, 3, 3)-4	1187840	13	(1, 2, 2, 2, 2, 2, 3)-5	0.88
32768	2893824	13	(1, 2, 2, 2, 2, 3, 3)-4	2539520	15	(1, 1, 2, 2, 2, 2, 2, 3)-6	0.88

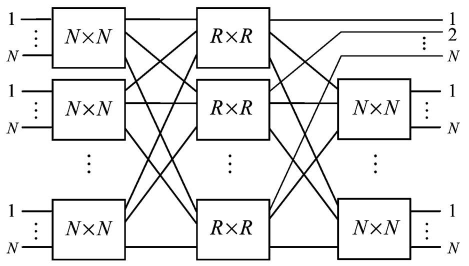

Fig. 11. A general construction of Waksman networks; note that the first switching element of the third stage is replaced by direct wiring.

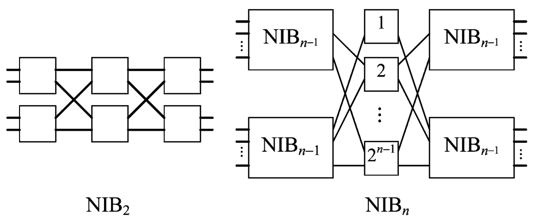

Fig. 12. Recursive construction of NIB_n (NIB_2 is equivalent to binary B_2).

4) $C(n, n+2, r)$ is NIR with input/output-divertability if input(output) switches are NIR with input/output-divertability and middle switches are RNB (by Theorem 5 and 9).

Since the above four kinds of networks are NIR with input/output-divertability, we then recursively use smaller networks to construct larger networks, and the larger networks are also NIR with input/output-divertability. Our algorithm works as follows.

Step 1) Set a single 2×2 crossbar as a *level*-1 network.
Step 2) Set NIB_2 and 2-B_2 as *level*-2 networks.
Step 3) For $i = 3$ to m (specified by user).
Step 4) Set NIB_i and 2-B_i as *level*-i networks.
Step 5) For $j = 2$ to $i - 2$.
Step 6) For each *level*-j network (remove one inlet or outlet), and for each *level*-$(i-j)$ network, use them to construct a *level*-i network by the $C(n, n+1, r)$ construction.
Step 7) For each *level*-j network (remove two inlets or two outlets), use B_{i-j} and the *level*-j network to construct a *level*-i network by the $C(n, n+2, r)$ construction (end of the for-loop of j).
Step 8) Sort the *level*-i networks by their *cost/size ratio* and output these networks (end of the for-loop of i).

This algorithm is used to generate as many NIR networks as possible by using our current knowledge, thus we are able to choose efficient NIR network of proper sizes. Since the generated NIR networks are of various sizes, we give the size-(cost/size) plot of the best seven networks of *level*-i networks, for size $\leq 100\,000$ (see Fig. 13). Note that, because we remove some inlets or outlets while applying $C(n, n+1, r)$ and $C(n, n+2, r)$ construction, most of the resulting networks are of distinct sizes.

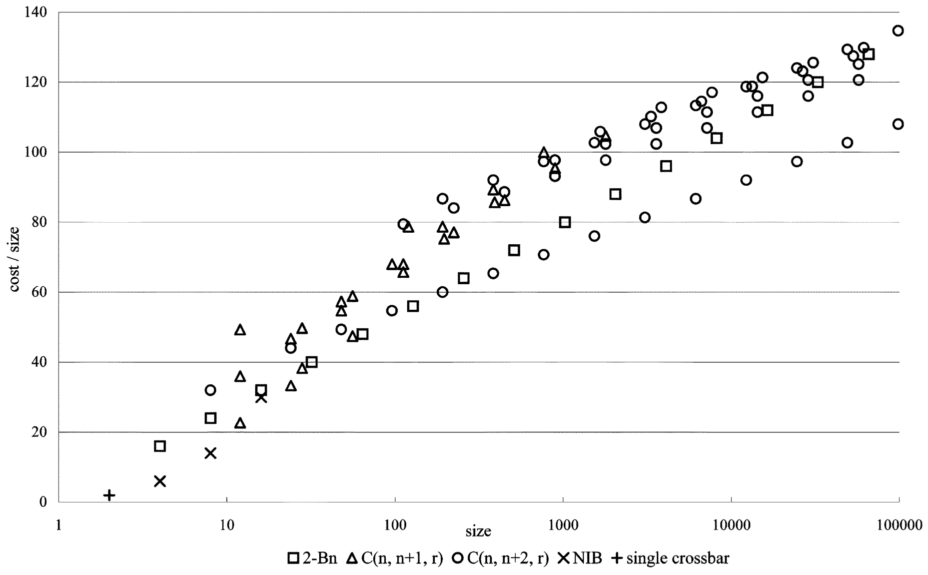

Fig. 13. Best seven networks of *level-i* networks, size ≤ 100000. Each point is marked by its last applied construction.

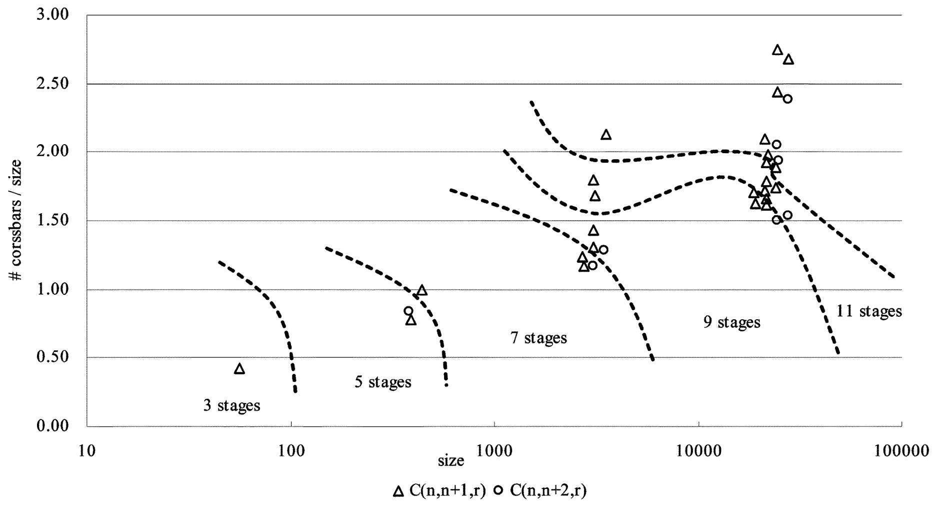

Fig. 14. Best nine networks of *level-i* networks for $d = 8$. Each point is marked by its last applied construction.

Thus, we compare them by the cost/size ratio, but not by cost. From Fig. 13, we find that the most cost-efficient NIR networks are (from small size to large size) NIB networks, networks applying $C(n, n+1, r)$ construction, 2-B_n, and networks applying $C(n, n+2, r)$ construction. It should be noticed that: 1) networks applying $C(n, n+2, r)$ construction are the most cost-efficient NIR networks for size greater than 512 and 2) 2-B_n and networks applying $C(n, n+2, r)$ construction require much fewer divertible crossbars than other networks.

C. *Only $d \times d$ Crossbars are Allowed*

As in Section VI-B, we use an algorithm to generate NIR networks using only $d \times d$ crossbars.

Step 1) Set a single $d \times d$ corssbar as a *level*-0 network.

Step 2) Set $C(d-1, d, d)$ using $d \times d$ crossbars as a *level*-1 network.

Step 3) For $i = 2$ to k (specified by user).

Step 4) Set $j = 0$.

Step 5) While $(2i+1) - 2 \times (2j+1) > 0$.

Step 6) Construct $C(n, n+1, r)(2i+1)$-stage networks by using *level-j* networks (of $2j+1$ stages) as input/output switches and *level*-$(i - 2j - 1)$ networks as middle switches. Set them as *level-i* networks.

Step 7) Construct $C(n, n+2, r)(2i+1)$-stage networks by using *level-j* networks (of $2j+1$ stages) as input/output switches and $(2 \times (i - 2j - 1) + 1)$-stage d-ary Benes networks as middle switches. Set them as *level-i* networks.

Step 8) Set $j = j+1$ (end of the while-loop and the for-loop of i).

The main differenct between this algorithm and the algorithm in last subsection is that: we replace some 2×2 crossbars by 1×2 (or 2×1) crossbars while using a smaller networks as input or ouput switches, and thus reduce some minor costs in last subsection, but we do not reduce any cost in this subsection—this is for mimicking design scenarios that only switching modules of the same size are available. Figs. 14 and 15 give the size-(number of crossbars/size) plots by assigning d as 8 and 4,

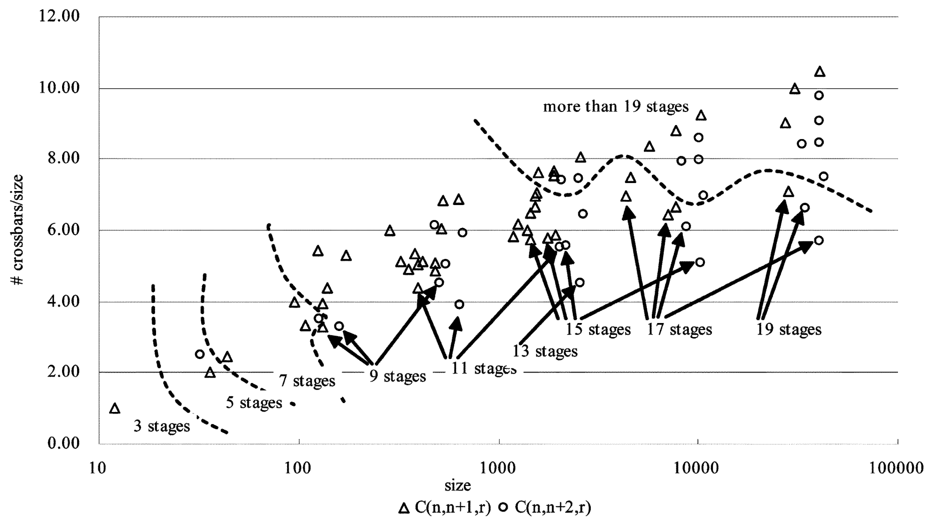

Fig. 15. Best nine networks of *level-i* networks for $d = 4$. Each point is marked by its last applied construction.

respectively. The best nine networks of each level are selected, and we show only networks of sizes $\leq 100\,000$. From Figs. 14 and 15, we find that larger networks applying $C(n, n+2, r)$ construction usually have better efficiency than other networks of the same stage numbers, but this trend is vanishing while using larger $d \times d$ crossbars.

REFERENCES

[1] V. E. Benes, *Mathematical Theory of Connecting Networks and Telephone Traffic*. New York: Academic, 1965.
[2] E. K. Bowdon, "Method and system for hitlessly rearranging connections in a cross-connect communications network," U.S. Patent 5,369,400, Nov. 29, 1994.
[3] F. K. Hwang, *The Mathematical Theory of Nonblocking Switching Networks*. Singapore: World Scientific, 1998.
[4] M. C. Paull, "Reswitching of connection networks," *Bell Syst. Tech. J.*, vol. 4, pp. 833–855, 1962.
[5] A. Waksman, "A permutation network," *J. ACM*, vol. 15, no. 1, pp. 159–163, 1968.

Frank K. Hwang received the B.A. degree from National Taiwan University, Taipei, Taiwan, R.O.C., in 1960, and the Ph.D. degree from North Carolina State University, Raleigh, in 1968.

He was with the Mathematics Center, Bell Laboratories, from 1967 to 1996. He is now a Chair-Professor with National Chiao Tung University, HsinChu, Taiwan, R.O.C. He has published approximately 350 papers and has authored or coauthored the following books: *The Steiner Tree Problem* (North-Holland, 1992), *Combinatorial Group Testing and Its Applications* (World Scientific, 1993, 2nd ed., 2000), *The Mathematical Theory of Nonblocking Switching Networks* (World Scientific, 1998), and *Reliabilities of Consecutive-k Systems* (Kluwer, 2000).

Wen-Dar Lin received a double B.S. degree in applied mathematics and computer science and information engineering, the M.S. degree in computer science and information engineering, and the Ph.D. degree in applied mathematics from National Chiao-Tung University (NCTU), HsinChu, Taiwan, R.O.C., in 1998, 2000, and 2003, respectively.

He then joined the Institute of Information Science (IIS), Academia Sinica, Taipei, Taiwan, R.O.C., as a Postdoctoral Fellow. At IIS, he is currently a member of the Bioinformatics Group in the Computer Systems and Communication Laboratory. His research interests include discrete mathematics, algorithms, group testing, and bioinformatics.

Vadim Lioubimov received the B.Sc. degree in mathematics from Kharkov National University, Kharkov, Ukraine, and the Ph.D. degree in mathematics from Concordia University, Montreal, QC, Canada, in 1998.

He was a Research Fellow with the Departments of Mathematics and Statistics with Concordia University and the University of Montreal. He is currently a Research Scientist with Almipa Inc., Canada, where he is developing video compression algorithms. He is also working independently on some mathematical research projects. His research interests include dynamical systems, algebraic combinatorics, discrete algorithms, and mathematical theory of switching networks.

Dr. Lioubimov is a recipient of a Natural Sciences and Engineering Research Council of Canada (NSERC) Postdoctoral Fellowship for 1999–2001.