On Nonblocking Properties of the Beneš Network

Petr Kolman*

Department of Applied Mathematics, Faculty of Mathematics and Physics,
Charles University, Prague, Czech Republic
kolman@kam.ms.mff.cuni.cz

Abstract. A network is called nonblocking if every unused input can be connected by a path through unused edges to any unused output, regardless of which inputs and outputs have been already connected. The Beneš network of dimension n is shown to be strictly nonblocking if only a suitable chosen fraction of $1/n$ of inputs and outputs is used. This has several consequences. First, there is a very simple strict sense nonblocking network with $N = 2^n$ inputs and outputs, namely a $(n + \log n + 1)$–dimensional Beneš network. Its depth is $O(\log N)$, it has $O(N \log^2 N)$ edges and it is not constructed of expanders. Secondly it leads to a $(3 \log N)$–competitive randomized algorithm for a $(\log N)$–dimensional Beneš network and a $O(\log^2 N)$–competitive randomized algorithm for a $(\log N)$–dimensional hypercube, for routing permanent calls.

1 Introduction

A network is a directed graph with two disjoint subsets of nodes called inputs and outputs. It is rearrangeable if, given any set of disjoint paths between some inputs and outputs, every unused input can be connected by a path through unused edges to any unused output with possible rearrangement of the previously established paths. If the rearrangeament is prohibited and still it is possible to find the path from an unused input to an unused output the network is called strict sense nonblocking.

Searching for disjoint paths is closely related to circuit-switching routing technique. This technique has been used in telecommunication networks, in parallel computers and also the ATM standart is based on virtual circuit routing.

The Beneš network has been proposed for use in telephone networks [6]. It and other closely related networks were also used as interconnection networks for parallel computers (e.g. BBN Butterfly GP1000). More recently, it has been suggested for use in ATM switches [16]. Beneš [6] and Beizer [5] proved that the Beneš network is rearrangeable, and Waksman [17] gave an simple recursive algorithm for establishing the connections. A natural question is whether the

---

* This research has been supported by the Grant Agency of the Czech Republic under grant No. 102/97/1055 and by European Community under project ALTEC-KIT INCO-COP 96-0195.

network is also strict sense nonblocking? We show that if we use more then a fraction of 3/4 of all inputs and outputs it is not. On the other hand if in n–dimensional Beneš network only a suitable chosen fraction of $1/n$ of inputs and outputs is used (e.g. the first $2^n/n$ inputs and outputs), then it is strict sense nonblocking. That is we have a very simple strict sense nonblocking network with $N = 2^n$ inputs and outputs, namely a $(n+\log n+1)$–dimensional Beneš network. It has $O(N \log^2 N)$ edges and depth $O(\log N)$. An almost identical network (the Cantor network, cf. [7]) was shown, by other method, to be strictly nonblocking by Melen and Turner [14].

There are assymptoticaly smaller strictly nonblocking networks. Bassalygo and Pinsker [4] proved existence of an assymptoticaly optimal network with $O(N \log N)$ edges (the lower bound was given by Shannon [15]). However, their result relies heavily on expander graphs. Also other constructions of nonblocking networks and their generalizations are based on expander graphs [1, 8]. If we compare the constants hidden in the big O notation in the result of Bassalygo and Pinsker [4] and in our result we find that for less than about 2^{15} inputs and outputs the Beneš network is the smaller one. Thus, from practical point of view the Beneš network is at least comparable.

This property has an application in on–line algorithms for call admission and circuit routing. It leads to a $(3 \log N)$–competitive randomized algorithm for the $(\log N)$–dimensional Beneš network and a $O(\log^2 N)$–competitive randomized algorithm for the $(\log N)$–dimensional hypercube, for permanent calls problem. Although there are algorithms with better competitive ratios for other networks (e.g. randomized logarithmically competitive algorithms for line [9], trees [2, 3] and meshes [11], deterministic $O(\log N)$–competitive algorithms for expander graphs [10]) we are not aware of any ones for Beneš network and the hypercube.

In the next section the crucial notion of i–similarity is introduced and some usefull lemmas as well as necessary and sufficient condition for existence of a free path in a partialy blocked Beneš network are given. This yields an efficient algorithm for searching for a path in a partialy blocked network in Sect. 3. The partially nonblocking property of the Beneš network and its consequences for the on–line routing are given in Sect. 4. Section 5 states some stronger results about Beneš networks of very small dimensions. Finally, in Sect. 6 it is shown that the Beneš network is not so good in the stronger nonblocking properties as it is in rearrangeability.

2 Definitions and Tools

A *network* is a directed graph with two disjoint subsets of nodes called inputs and outputs. Its *depth* is the lenght of the maximal shortest path between inputs and outputs, and its *order* is the number of its inputs and outputs. There are two types of *requests* for a network: (i) to make a connection between given input and output, and (ii) to destroy a connection. If not stated otherwise, connection requests will be considered only between unused inputs and outputs.

Most of the time we will consider blocking of edges. Therefore the following definitions are given only for this case. However, when needed (Theorem 10) analogous definitions will be assumed also for a case of blocking nodes.

A network with N inputs and N outputs is called *strict sense edge nonblocking of order N* [1] if, no matter what paths are established in it, it is possible to connect every unused input to any unused output through unused edges. If there is an algorithm that is able to serve every sequence of requests in an edge disjoint fashion, starting from an empty network, the network is called *wide sense edge nonblocking* [8, 1]. If there is an algorithm that is able to serve each sequence of connection requests only we call the network *weak sense edge nonblocking.* A network is *edge rearrangeable* [6, 1] if for any set of requests given at once it is possible to serve all the requests in an edge disjoint fashion. Clearly, any strict sense nonblocking network is wide sense nonblocking, wide sense nonblocking is weak sense nonblocking, weak sense nonblocking is rearrangeable.

The n-dimensional *butterfly* network is defined as follows [12, p.454]. The nodes correspond to pairs (w, i), where i is an integer, $0 \leq i \leq n$, and w is an n-bit binary number. Usually i is called the *level* of the node and w its *row.* There is an edge from (w, i) to (u, j) if and only if $j = i + 1$, and w and u are either identical, or they differ precisely in the jth bit. The n-dimensional *Beneš* network B_n consists of two back-to-back butterflies. Overall it has $2n + 1$ levels. The 0–level nodes are the inputs, $2N$–level nodes are the outputs.

Numbers a and b are said to be *i–similar* if their binary representations agree on i least significant bits. Their *similarity* is 2^{i-1}. Two requests or paths from a to b and from c to d are *i–similar* if either a and c, or b and d are i–similar.

Lemma 1. *Suppose there are k paths established in the Beneš network B_n. Then these paths are edge disjoint if and only if there is no Beneš subnetwork B_i, $0 \leq i \leq n$, and no paths p_1, p_2 such that both p_1 and p_2 are passing through the B_i and they are $(i + 1)$–similar.*

Proof. $\Rightarrow$ By contradiction. Suppose the paths are edge disjoint and there is B_i with two paths p_1 and p_2 passing through it, with their *outputs*, without loss of generality, $(i + 1)$–similar. Then, after $n + i$ steps they have to reach a node on a line that is i–similar with their outputs and, since in each B_i there is only one such line, the paths reach the same node after $n + i$ steps. But because the outputs are $(i + 1)$–similar both paths have to continue via the same edge from the B_i, which is a contradiction.

$\Leftarrow$ Analogically. □

Lemma 2. *Suppose there are k paths established in the Beneš network B_n and there is a request for the next path from a to b. Then a path for this request cannot go, in an edge disjoint fashion with the previously established paths, through a vertex u on the medium level n if and only if there is an already established path p such that*

- *p is $(i + 1)$–similar with the request $a \to b$, and*
- *p is passing through a medium level vertex v such that $\lfloor u/2^i \rfloor = \lfloor v/2^i \rfloor$.*

Proof. $\Rightarrow$ Let us establish the path from a to b through u. Then it will use some edge already used by some path p. Let i be maximal number such that p and $a \to b$ are $(i+1)$–similar. Then both of them pass through the same B_i (Lemma 1), and thus $\lfloor \frac{u}{2^i} \rfloor = \lfloor \frac{v}{2^i} \rfloor$, i.e. we have the desired path.

$\Leftarrow$ The path p disables all medium level nodes in the B_i it passes through from being a medium level nodes for path from a to b. Since $\lfloor \frac{u}{2^i} \rfloor = \lfloor \frac{v}{2^i} \rfloor$, we know u is one of these vertices. ☐

The lemma says that each path p, which is $(i+1)$–similar with the given request $a \to b$, disables the interval $I_p = \langle k2^i, \ldots, (k+1)2^i - 1\rangle$ of 2^i medium level nodes from being a medium level node of the path from a to b, where $k = \lfloor \frac{v}{2^i} \rfloor$ and v is the medium level node of p (Fig. 1). We have the following theorem:

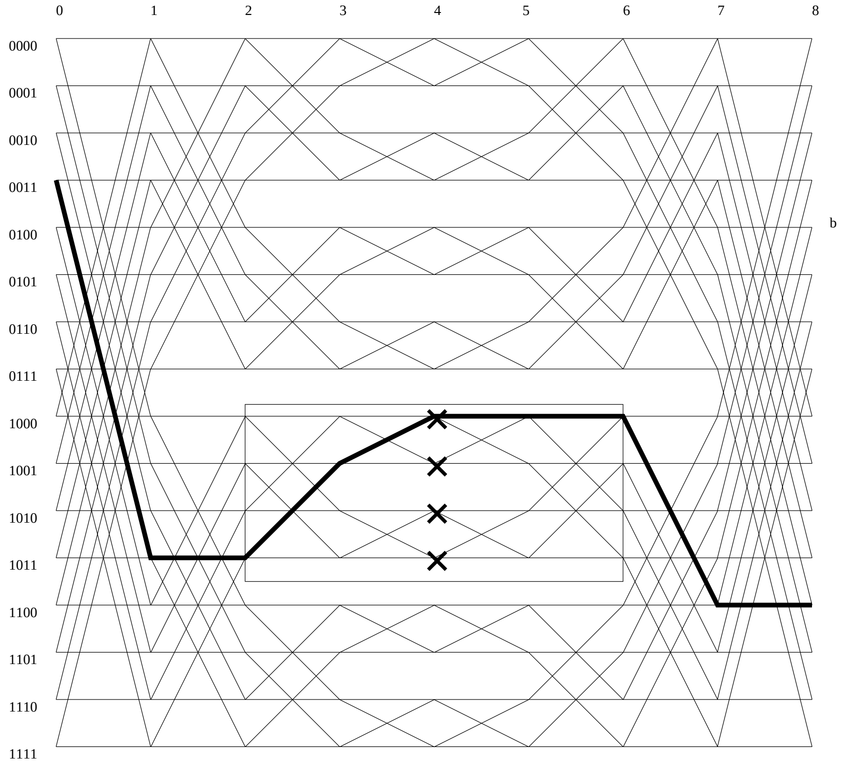

Fig. 1. The path p from input 3 to output 12 disables medium level nodes $8, 9, 10, 11$ from being a medium level node for a path from a ($0 \leq a \leq 15, a \neq 3$) to $b = 4$. The path p and The request $a \to b$ are (at least) 3–similar, their similarity is $2^{3-1} = 4$, $k = 8/4 = 2$, $I_p = \langle 8, 9, 10, 11\rangle$.

Theorem 1 (Necessary and sufficient condition for existence of a free path). *Given a set P of paths and a request for a path from input a to output*

b, there is a free path from a to b if and only if

$$\bigcup_{p \in P} I_p \subsetneq \langle 0, 2^n - 1\rangle \quad .$$

Let us notice also that for two intervals I_p and I_q, corresponding to paths p and q, such that $I_p \cap I_q \neq \emptyset$ and $|I_p| \leq |I_q|$, it is always the case that $I_p \subseteq I_q$.

3 Searching for a Path

Theorem 2. *Given a set P of k paths in the Beneš network and a request for a path from input a to output b, it is possible to decide in time $O(k)$ whether there exists a free path for this request, and if it exists to describe it.*

Proof. Due to Theorem 1 it suffices to decide whether the union of intervals I_p covers all nodes on the medium level or not. We shall use an array c of size 2^n to solve the problem. For simplicity we shall assume that this array is initialized to 0, later on we shall see how to avoid this requirement (we would like to avoid it since the initialization itself would take $\Omega(2^n)$ time). By taking any of the uncovered nodes in the medium level, if there is any, as the medium level node for the path from a to b we have also unique description of the new path.

Algorithm

```
for each path $p \in P$ do
    $i = \max\{similarity(a, in_p), similarity(b, out_p)\}$
    if $i > 0$ then
        $k = \lfloor medium_p / 2^i \rfloor$
        $b_p = k2^i$                  /* beginning of the interval $I_p$ */
        $e_p = (k+1)2^i - 1$          /* end of the interval $I_p$ */
        $c[b_p] = max\{e_p, c[b_p]\}$
    endif
enddo
$pointer = 0$
while ($pointer < 2^n$) and ($c[pointer] \geq pointer$) do
    $pointer = c[pointer] + 1$
enddo
if $pointer < 2^n$ then “There is a free path via medium level node $pointer$.”
    else “There is no free path from $a$ to $b$.”
endif
```

The values in the array c have the following properties at the end of each round of the first cycle: if $c[j] = 0$ then j is not a lower endpoint of interval I_p for any path p considered so far, and if $c[j] \neq 0$ then $c[j]$ is the upper endpoint of the largest interval with lower endpoint in j. It follows from this and from the fact that one of two overlapping intervals is always contained in the other that at the end of the algorithm *pointer* points to the minimal index unblocked medium level node, if there is any, or $pointer = 2^n$ if there is none.

It remains to show how to avoid the initialization. We will add some information to the array c: whenever we set the value of $c[j]$ we add information about due to which path we set $c[j]$ in this way. Whenever reading $c[j]$ we verify whether the information read is really valid. The case $c[j]$ is not valid corresponds to $c[j] = 0$. This change causes only a constant slowdown.

Since both cycles run at most k times and each round takes $O(1)$ time, (in our model we count for an operation on two n–bits numbers a unit cost) the algorithm runs in time linear in the number of paths. □

For establishing and searching for paths it is also possible to use a parallel breadth first search with running time $O(n)$.

4 Nonblocking Property of the Beneš Network and its Application for On–line Routing

Theorem 3. *The Beneš network B_n is a strict sense edge nonblocking network of order $2^n/n$.*

Proof. Let us take as inputs only the 0–level nodes that agree on the $\lceil \log n \rceil - 1$ most significant bits (the leftmost) and similarly as outputs only the $2n$–level nodes that agree on the $\lceil \log n \rceil - 1$ most significant bits. The number of such inputs (and outputs) is $2^{n-(\lceil \log n \rceil - 1)} \geq 2^n/n$. It does not matter what is the actual setting of the $\lceil \log n \rceil - 1$ bits, especially the setting of the inputs need not be the same as of the outputs.

Suppose there are some paths established between the chosen inputs and outputs. Consider any request $a \to b$ from an unused input a to an unused output b. Our aim is to prove an existence of a free path for it. The proof will rely on the Theorem 1. We will show that the sum of similarities of a with all other input nodes is strictly less than 2^{n-1}, and the same for b and output nodes: The number of i–similar and not $(i+1)$–similar inputs is $2^{n-i-\lceil \log n \rceil}$, for $1 \leq i \leq n - \lceil \log n \rceil$. The similarity of each such input is 2^{i-1} and the sum of similarities of all i–similar inputs is $2^{n-\lceil \log n \rceil - 1}$. Since $(n - \lceil \log n \rceil)2^{n-\lceil \log n \rceil - 1} < 2^{n-1}$ and the same holds also for the outputs, we conclude that the sum of similarities of the current request with all already established paths is strictly less then 2^n. So the union $\bigcup_{p \in P} I_p$ of intervals is surely a proper subset of $\langle 0, 2^n - 1 \rangle$ and there is a free path from a to b. □

Consider the following modification of the $(n + \lceil \log n \rceil)$–dimensional Beneš network: replace each path of lenght $\lceil \log n \rceil$ starting in the first 2^n inputs by a single edge, and remove all other edges in the first $\lceil \log n \rceil$ levels. Do the same for outputs. You will get a Cantor network and it was also proved by other method to be strictly nonblocking [14].

From this point to the end of the section a slightly different kind of requests will be considered. There will be no destruction requests but only requests for connection. Each request will contain information about its starting and final points, which have to be 0–level and $2n$–level nodes, and about its duration. If the duration of all calls is infinite we speak about permanent calls problem, otherwise about known limited duration call problem. There is no restriction on the number of requests originating and terminating in a node. The property of the Beneš network described in Theorem 3 leads to a randomized $O(n)$-competitive on-line algorithm for the n-dimensional Beneš network and similar algorithm for the hypercube. We analyze our algorithms against oblivious adversary. It's worth noting than even only weak sense nonblocking property will yield the same results.

Theorem 4. *There is a randomized $3n$–competitive algorithm for permanent calls problem on the n-dimensional Beneš network.*

Proof. The immediate consequence of Theorem 3 is that there is a deterministic 3–competetive algorithm for establishing paths between only the first 2^n inputs and outputs on a $(n + \log n)$–dimensional Beneš network. Consider any such algorithm A. Choose uniformly at random an integer r, $0 \leq r < n$. Accept only those requests that are routed by A through any of medium level nodes $r2^n, r2^n + 1, \cdots, (r+1)2^n - 1$ and route them on B_n in the same way as they are routed by A through the $2n$ medium levels of the higher dimensional network. The expected number of calls accepted by the described algorithm is at least $1/(3n)$ of all calls and all of them are routed via edge disjoint paths. ☐

Because of similarities of hypercube and Beneš network the algorithm can be extended also for the hypercube, at the expense of a higher power of n.

Theorem 5. *There is a randomized $O(n^2)$-competitive algorithm for permanent calls problem on the n-dimensional hypercube.*

Proof. For simplicity a directed hypercube with *doubled* edges HD_n will be considered. However, this assumption will be used in such a way that it may be easily removed by increasing the dimension of the hypercube by one. Consider the following embedding of n-dimensional Beneš network into HD_n: all nodes (w, i), $0 \leq i < 2n$, from raw w are mapped to a node w of the hypercube. Given the embedding, simulate the algorithm for Beneš network on the hypercube. The higher power of n is due to the fact that this algorithm is garanted to accept only one call from and to a node, resp., whereas the optimal algorithm might accept up to n calls from and to the node, resp. ☐

The algorithms can be extended for known limited duration calls [2, 13]. The competitive ratio is multiplied by a factor of $\log T$ ($\log T \log^{1+\epsilon} \log T$, resp.) where T is the known (unknown, resp.) bound on the ration between the minimum and maximum duration of calls.

5 Small Dimension Beneš Networks

In this section some new notation is used. Let $B_1(p)$ denotes the Beneš subnetwork B_1 that p is passing through, and $\neg B_1(p)$ the other B_1 belonging to the same B_2. We say that two paths p, q such that $B_1(p) = B_1(q)$ are *parallel* if they do not have a common vertex in the $B_1(p)$. When it is allowed for two requests to start in the same node then we speak about $2-1$ requests. Moreover, when two requests may terminate in the same node they are called $2-2$ requests.

Theorem 6. *The Beneš network* B_2 *is a wide sense edge nonblocking network for* $2-1$ *requests.*

Proof. Our algorithm makes routing decisions according to five rules:

1. Whenever routing a second request through a B_1, do it in a non-parallel way.
2. If there is no path in the B_2 then route the request through the upper B_1.
3. If there is only one path p in the B_2 then route the request through $B_1(p)$ if possible.
4. If there are two paths p, q then route the request through $B_1(p)$ if possible.
5. Suppose there are three paths p, q, s and it is possible to route the request both through the upper B_1 and the lower B_1. Without loss of generality we may assume that $B_1(p) = B_1(q)$. If by routing the request through $B_1(p)$ no new parallel paths would emerge then route the request through the $B_1(p)$, otherwise through $\neg B_1(p)$.

The rules 2-5 describe decisions about the first step, rule 1 about the second.

Consider a configuration of paths in B_2. If there exist paths c_1, c_2, c_3 such that c_1, c_2 are parallel, and c_3 could pass in a non-parallel way through $B_1(c_1) = B_1(c_2)$ but does not, we call it a *dead configuration*. Two lemmas will be proved that together show the correctness of the algorithm.

Lemma 3. *If the already established paths are not in a dead configuration, then the algorithm is able to route the next request.*

Lemma 4. *The algorithm never establishes a dead configuration.*

Proof. (Lemma 3) There is always enough space for routing a request if there are at most two established paths. Therefore assume there are three paths p, q, s in the B_2 not in a dead configuration, and a request $a \to b$ is given. If the input a was not used up to this time, then there is a free path from a to b either through the upper or lower B_1 and the algorithm finds it. The situation is a bit more difficult if a is already used by a path, say by p, since then the request cannot go through $B_1(p)$. If at most one of q, s is passing through $\neg B_1(p)$, or if both q, s are passing through $\neg B_1(p)$ and they are not parallel, then there is a free path from a to b through $\neg B_1(p)$ and the algorithm finds it. The only remaining case is that q and s are parallel and $B_1(p) \neq B_1(q) = B_1(s)$. Assume for a while that q and s are the only paths in B_2 and consider an unused input u. Then u may be connected with exactly one of the two unused outputs by a path passing through $B_1(q)$. Now consider the path p again. From the observation above and from the fact that p, q, s are not in a dead configuration it follows there is free path for the request through $\neg B_1(p)$ and the algorithm finds it. $\neg B_1(p)$ and the algorithm finds it.

Proof. (Lemma 4) From the description of the algorithm it follows that the property is always satisfied whenever a third path is established. Also deleting any path will not create a dead configuration. Suppose now there are three paths not in a dead configuration and a request $a \to b$ is given. There are two ways how the "not dead" property may be broken: by establishing the path c_3 or by establishing c_1 or c_2 (i.e. one of the two parallel paths). By rules 1. and 5. we get that no parallel path will emerge if there is a choice for the first edge of a path from a to b, or if there is no choice for it but there is at most one other path in the B_1 that the request is forced to go through. Neither it happens in these cases that a path c_3 emerges. The only left case is that the request is forced to pass through B_1 with two already established paths in it. Let p be the path that forces the request to do so, that is let p be the path that starts in a. The assumption that the configuration was not dead and the fact that the path p and the request have to use different B_1 yield that neither in this case a dead configuration is created.

□

Theorem 7. *The Beneš network B_3 is a wide sense edge nonoblocking network for $1-1$ requests.*

Proof. Consider this algorithm. If the endpoint of the request is in the lower half of B_3 then use for the path as the first edge the edge to the lower B_2, otherwise to the upper. By doing so we split the problem into two $2-1$ routing problems on B_2 which we will deal with according to the Theorem 6. □

Corollary 1. *There is a deterministic 3–competetive algorithm for permanent routing problem on a 3–dimensional Beneš network.*

Theorem 8. *The Beneš network B_3 is a weak sense edge nonoblocking network for $1-1$ requests.*

As we noticed at the beginning this is an immediate consequence of the previous theorem. We state it separately because of a very simple direct proof of it. Moreover, our conjecture is that the algorithm described in this proof works for the Beneš network B_4 as well.

Proof. Consider a greedy algorithm: route the given request through as low medium node as possible. With a little of reasoning it is possible to show that the algorithm is always able to establish all 8 paths. □

6 Negative Results

Theorem 9. *The Beneš network B_n is not a wide sense edge nonblocking network of order 2^n for $2-2$ requests, for $n \geq 2$.*

Proof. We have to show that there is no algorithm for establishing the paths when the requests are presented on-line. Suppose, by contradiction, there is such an algorithm. We will describe a sequence of requests such that it is unable to establish the edge disjoint paths for them. Let a, c be two different inputs and b, d, e three different outputs. Consider the following sequence of reguests:

1. $a \to b$, let suppose, w.l.o.g., that it passes through the lower B_{n-1},
2. $a \to e$, it has to pass through the upper B_{n-1}
3. $c \to d$, if it passes through the lower B_{n-1}, then there is no place left for request $c \to e$; if it passes through the upper B_{n-1}, then there is no place left for $c \to b$.

It is easy to see that on B_1 this argument cannot be used and that B_1 is a wide sense edge nonblocking network of order 2 for $2-2$ requests. □

Theorem 10. *The Beneš network B_n is not a wide sense node nonblocking network of order 2^n for $1-1$ requests, for $n \geq 3$.*

Proof. Again a sequence of requests will be provided that no deterministic algorithm is able to serve. Consider the following sequence of requests, for $N = 2^n$:

1. $a \to b$, $a \geq 5N/8$, $b \geq 3N/4$, let suppose, w.l.o.g., that it passes through the lower B_{n-1},
2. $c \to d$, $d = b - N/4$, if it would go through the upper B_{n-1} then there is no place left for $x \to y$, $x = a - N/2$, $y = d - N/2$, so let us suppose it passes through the lower B_{n-1},
3. $e \to f$, $e = a - N/8$, $f = b - N/2$, it has to go through the upper B_{n-1}, since both nodes in the level $2n-2$ of the lower B_{n-1}, from which the output f is reachable, are already blocked,
4. $g \to h$, $g = a - 5N/8 = e - N/2$, $h = d - N/2$ – there is no place left for it.

□

Theorem 11. *The Beneš network B_n is not a strict sense edge nonblocking network of order 2^n for $1-1$ requests, for $n \geq 3$.*

Proof. For simplicity the proof is described only for B_3. Consider the following six paths (b_i says whether the i-th edge on the path leads to the upper ($b_i = 0$) or lower ($b_i = 1$) Beneš subnetwork B_{n-i}).

input	1	2	3	4	6	7
output	2	3	6	5	1	4
$b_1b_2b_3$	000	011	010	101	001	111

Then it is impossible to establish an edge disjoint path for request $0 \to 0$. □

Corollary 2. *The Beneš network B_n, $n \geq 3$, is not a strict sense edge nonblocking network of order s, for any $s > \frac{3}{4}\, 2^n$.*

Proof. If we allow more then 6 out of any 8 $(n-3)$–similar 0–level nodes to be input nodes and more then 6 out of any 8 $(n-3)$–similar $2n$–level nodes to be output nodes, then the construction from Theorem 11 (or a similar one) shows the network is not a strict sense nonblocking. □

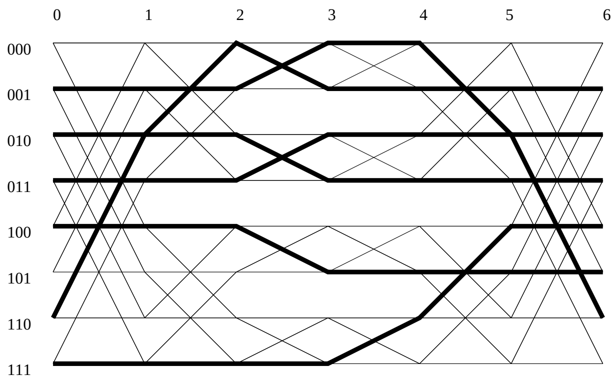

Fig. 2. The Beneš network B_n is *not* a strict sense edge nonblocking network of order 2^n for $1-1$ requests. The six described paths disable establishing a path from 0 to 0.

7 Open Problems

There are still many open problems about the Beneš network and the hypercube. Let us mention some of them.

- We know B_n is not strict sense nonblocking network of order larger than $3 \cdot 2^n/4$, but it is one of order $2^n/n$. Is B_n strict sense nonblocking network of larger order?
- Of what order is Beneš network wide sense nonblocking? Is it higher than for strict sense nonblocking?
- What are the optimal competitive ratios for randomized algorithms for permanent calls on Beneš and hypercube networks?

8 Acknowledgements

The author would like to thank Jiří Sgall for many valuable discussions, and also the anonymous rewiever for pointing to the paper of Melen and Turner [14].

References

[1] S. Arora, F. T. Leighton, and B. M. Maggs. On-line algorithms for path selection in a nonblocking network. *SIAM Journal on Computing*, 25(3):600–625, June 1996.

[2] B. Awerbuch, Y. Bartal, A. Fiat, and A. Rosén. Competitive non-preemptive call control. In *Proceedings of SODA*, pages 312–320, 1994.

[3] B. Awerbuch, R. Gawlick, T. Leighton, and Y. Rabani. On-line admission control and circuit routing for high performance computing and communication. In *Proceedings of FOCS*, pages 412–423, 1994.

[4] L. A. Bassalygo and M. S. Pinsker. Complexity of an optimum nonblocking switching network without reconnections. *Problems of Information Transmission (translated from Problemy Peredachi Informatsii (Russian))*, 9:64–66, 1974.

[5] B. Beizer. The analysis and synthesis of signal switching networks. In *Proceedings of the Symposium on Mathematical Theory of Automata*, pages 563–576. Brooklyn, NY, Brooklyn Polytechnic Institute, 1962.

[6] V. E. Beneš. Permutation groups, complexes and rearrangeable connecting network. *The Bell System Technical Journal*, 43, 4:1619–1640, 1964.

[7] D. G. Cantor. On non-blocking switching networks. *Networks*, 1:367–377, 1972.

[8] P. Feldman, J. Friedman, and N. Pippenger. Wide-sense nonblocking networks. *SIAM Journal on Discrete Mathematics*, 1, 1988.

[9] J. A. Garay, I. S. Gopal, S. Kutten, Y. Mansour, and M. Yung. Efficient on-line call control algorithms. In *Proceedings of the 2nd Israeli Symposium on Theory of Computing and Systems*, pages 285–293, 1993.

[10] J. Kleinberg and R. Rubinfeld. Short paths in expander graphs. In *Proceedings of FOCS*, pages 86–95, 1996.

[11] J. Kleinberg and É. Tardos. Approximations for the disjoint paths problem in high-diameter planar networks. In *Proceedings of STOC*, pages 26–35, 1995.

[12] F. T. Leighton. *Introduction to Parallel Algorithms and Architectures: Arrays - Trees - Hypercubes.* Morgan Kaufmann, San Mateo, 1992.

[13] R. J. Lipton and A. Tomkins. Online interval scheduling. In *Proceedings of SODA*, pages 302–311, 1994.

[14] R. Melen and J. S. Turner. Nonblocking multirate networks. *SIAM Journal on Computing*, 18(2):301–313, 1989.

[15] C. E. Shannon. Memory requirements in a telephone exchange. *Bell Syst. Tech. Journal*, 29:343–349, 1950.

[16] J. Turner and N. Yamanaka. Architectural choices in large scale atm switches. Technical Report WUCS-97-21, Department of Computer Science, Washington University Saint Louis, 1997.

[17] A. Waksman. A permutation network. *Journal of the ACM*, 15(1):159–163, January 1968.