University of Castilla–La Mancha

A publication of the

Computing Systems Department

An Alternative for Building High-Radix Switches: Formalization and Configuration Methodology*

by

J.A. Villar, F.J. Andújar, J.L. Sánchez, F.J. Alfaro, J. Duato

Technical Report **#DIAB-11-02-1** February 2011

(*) This work was supported by the Spanish MEC and MICINN as well as European Comission FEDER funds, under Grants "Consolider Ingenio–2010 CSD2006-00046" and "TIN2009–14475–C04", respectively; it was also partly supported by Junta de Comunidades de Castilla–La Mancha under grants "PCC08–0078–9856" and "POII10–0289–3724".

DEPARTAMENTO DE SISTEMAS INFORMÁTICOS
ESCUELA SUPERIOR DE INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE CASTILLA-LA MANCHA
CAMPUS UNIVERSITARIO s/n
02071, ALBACETE, ESPAÑA
Tlf. +34.967.599200, Fax +34.967.599224

An Alternative for Building High-Radix Switches: Formalization and Configuration Methodology

Juan A. Villar, Francisco J. Andújar
Instituto de Investigación en Informática
Campus Universitario s/n
02071 – Albacete, España
{juanan,fandujar}@dsi.uclm.es

Francisco J. Alfaro, José L. Sánchez
Departamento de Sistemas Informáticos
Escuela Superior de Ingeniería Informática
02071 – Albacete, España
{falfaro, jsanchez}@dsi.uclm.es

José Duato
Dpto. de Ingeniería de Sistemas y Computadores
Camino de Vera, s/n
Universidad Politécnica de Valencia
46022 – Valencia, España
jduato@gap.upv.es

February 7, 2011

Contents

1 Introduction

Interconnection networks are a key component for a wide range of multiprocessor systems, ranging from large supercomputers to multicore chips. High performance networks are essential in these systems, where high reliability in communications, high information transfer rates and very low latencies are critical. Often, the interconnection network is the subsystem that a more custom design requires. For instance, Tianhe-1A supercomputer [SS10], number one in the November 2010 Top500 list, is composed of standard Intel® and NVIDIA® processors and a fancy new interconnection network. This custom interconnect design removes the interconnect bottleneck and significantly contributes to the high global performance of Tianhe-1A.

Interconnection network design is determined by the available technology. Recent advances on the technology have substantially improved the performance of the basic network components: links and switches. The latter are responsible for most of the interconnection network performance, and so they are the subject of major research. One of the main parameters characterizing network switches is the number of ports, which has a strong influence on cost, consumption and performance in the whole system.

Given a multiprocessor system with a large number of connected elements, increasing the number of switch ports results in a decrease in the number of switches and network links. As the cost of the network is proportional to the number of switches, it is clear that cost decreases by using switches with higher number of ports. Moreover, total consumption of the network is also considerably reduced as it is directly proportional to the number of switches in the network.

Regarding performance, in terms of latency, for example, it is clear that the use of switches with more ports involves a reduction in the average time to transfer data over the network. In particular, using fewer switches to connect the same number of elements reduces the number of hops and the number of possible packet collisions in the network, and thus the time to reach their destinations. Furthermore, having less switches, the total processing time of the packets in the switches along their paths is also reduced.

Thus, the design of switches with a high number of ports is an attractive option to improve the performance and reduce the cost of the interconnection network, specially for large multiprocessor systems. However, there are some problems to design such switches. One of them is related to the complexity of the switch logic. The switch becomes more complex as radix increases, taking up to a significant percentage of total system power [WPM03]. The balance between cost and efficiency is not easy to work through, requiring a deep study regarding this trade-off. On the one hand, the size of some switch structures grows quadratically with the number of ports. That is the case of, for example, the aggregate buffer requirements as identified in [GAG+03], or the schedulers as stated in [MAM+05]. Moreover, traditional flow control policies are also affected by switch radix in two aspects [MG07]: the round trip time drastically increases, and the memories for storing flow control credits are linearly dependent on

the round trip time. On the other hand, pin count will slowly increase next decade, according to the ITRS [fSI10], and therefore switch ports number will slightly increase. Moreover, there are difficulties to apply some improvement techniques when the number of ports is high. For instance, Virtual Output Queuing (VOQ) implementation becomes unfeasible in practice for switches with large number of ports. To overcome these problems, different solutions have been proposed, but actually, they are postponing the problem for coming switch generations.

In any case, switch size constraints are mainly determined by the current integration scale and package pin count. To go beyond the integration scale bounds, an alternative solution for building high-radix switches is the combination of several low-radix switches. This solution has the advantage of obtaining higher number of ports. Moreover, some difficulties mentioned above lose importance.

The main idea is to implement n-port switches from several smaller m-port switches. For instance, a n-port switch consisting of two identical m-port switches ($n/2 < m < n$) can internally interconnect each other using $m - n/2$ ports, using the remaining ports for external connections. Note that this strategy will remain valid as integration scale keeps evolving.

An important consequence of this strategy for building larger switches is that the resulting switch will no longer be homogeneous. Switch performance will vary depending on the internal configuration. The internal switches interconnection can become a potential bottleneck if they have to support most of the traffic handled in the switch. Therefore, it is essential to minimize the impact of this bottleneck, otherwise the latency on the network will be increased. Thus, the switch-level connection pattern[1] becomes an important design decision in the construction of this kind of switches. An arbitrary pattern may produce a significant performance degradation. Consequently, it is necessary to determine the most convenient pattern in order to obtain the best switch performance.

In this paper, we present a formalization of this kind of high-radix switches and propose a methodology for configuring them in an optimal manner when they are used to build large switch–based interconnection networks. To show how this methodology works, it is applied to a particular interconnection network.

The report is organized as follows: Section 2 describes the *combined switches*, and in Section 3 we introduce the notation used later. Section 4 formally defines and characterizes the *twin switches*. In Section 5, we propose our methodology for searching the optimal configuration of *twin switches*, and show how it works in a particular case. Section 6 gives a brief review to existing proposals on high-radix switches. Additionally, in the A we have included basic concepts on multistage interconnection networks.

[1] We distinguish between *network-level connection pattern* and *switch-level connection pattern*. The former is the traditional interconnection pattern connecting switch-based networks (e.g. *butterfly* permutation in multistage interconnection networks); and the latter refers to how every high-radix switch ports are mapped to the ports of the internal switches.

2 High-Radix Switches by Combining Low-Radix Switches

As mentioned above, it is possible to build high-radix switches by combining several low-radix switches. This strategy makes possible to eventually overtake integration technology and dramatically shorten the time-to-market. Note that this will remain valid as integration technology continues evolving.

This strategy opens a series of new issues that must be addressed so that it would be implemented in an efficient manner. In this section, we define the combined switches and briefly overview the issues characterizing them. Then in the next sections, we formally analyze them in depth.

2.1 Combined Switches

In this section we define the combined switches giving a general definition. Then we turn our attention to a particular subclass of this kind of switches, which will be used in order to show the characteristics and evaluate the performance of combined switches as a high-radix switch alternative.

Definition 1.1 *A Combined switch, or simply C-switch, is a switch formed by several smaller interconnected switches (internal switches). The ports being offered by a C-switch are obtained from free ports of its internal switches after they are interconnected.*

This is a general definition because it does not specify either the number of internal switches or their radix. Therefore any switch obtained by combining other lower switches is included in this category. However, there exist some difficulties to build C-switches having many internal switches and a high heterogeneity degree.

In order to keep low the internal latency, a full-connected subnetwork interconnecting all the internal switches is preferred. As the number of internal switches increases, the number of ports dedicated to subnetwork connections grows as fast as the number of ports devoted to external communications decreases, so this way of building high-radix switches would lose interest. Therefore, it seems reasonable that the number of internal switches may not be too large.

On the other hand, a simpler C-switch internal design can be achieved if all the internal switches are equal. Although this aspect is not as restrictive as the number of internal switches, it would be also recommendable that all the internal switches have the same radix.

An interesting case is that where C-switches are built from only two identical internal switches. This subclass of C-switches still offers an important increase in the

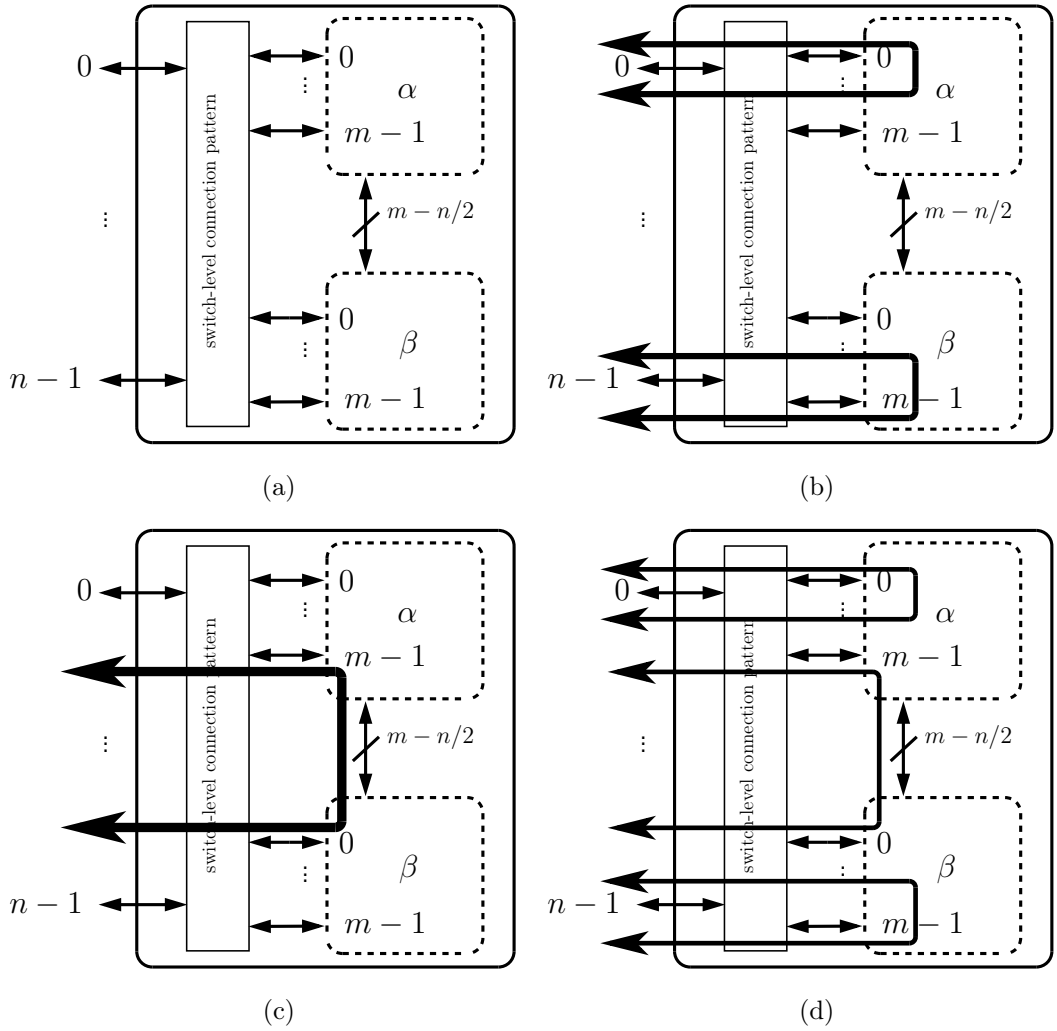


Figure 1: (a) Basic block diagram of a T-switch and several cases: (b) Optimal, (c) Bad, and (d) General.

number of ports while the interconnection between the two internal switches is the simplest one.

Definition 1.2 *A Twin switch, or simply T-switch, is a switch formed by two identical smaller interconnected switches. The ports being offered by a T-switch are obtained from free ports of its two internal switches after they are interconnected.*

Considering that the two internal switches and the T-switch have radixes m and n, respectively, Figure 1(a) shows a basic diagram of a T-switch, where internal switches are denoted as α and β. Although T-switches seem to be simple, there are significant challenges in its design. Two of them stand out especially: (1) to obtain the appropriate switch-level connection pattern of internal subnetwork, (2) to determine the adequate number of ports used to interconnect switches α and β.

Switch-level connection pattern has an important influence on packet latency. Time to cross the T-switch will be minimal when only one internal switch (α or β) is

used (Figure 1(b)). The bad case is obtained when both internal switches are used for every path crossing the T-switch (Figure 1(c)). To obtain the best case is not trivial and an in-depth study is required, in which several factors, e.g., network topology, routing and traffic pattern must be considered. From Section 2.2 we show in a formal way how the optimal switch-level connection pattern can be obtained if these factors are considered.

Regarding the second challenge, the number of internal ports must be that which avoids the creation of the internal bottleneck between α and β. Obviously, the number of internal ports and the switch-level connection pattern have a clear interdependence.

The situations shown in Figure 1(b) and Figure 1(c) are appropriate for illustrating the T-switch design challenges, but a more common situation is that shown in Figure 1(d), where both kinds of cases coexist. In such situations, the main objective in the switch-level connection pattern design of T-switches is to minimize the use of the ports interconnecting the internal switches.

Therefore, since in some cases communication will require the use of both internal switches (i.e. a path passing through a T-switch will use both α and β), we have to avoid interconnection between α and β becomes a bottleneck. Moreover, the adequate number of ports of each internal switch to connect with each other must be determined. It is clear that the greater the number of ports used to interconnect the internal switches the lower the probability of this interconnection becomes a bottleneck. However, as mentioned above, as the number of ports devoted to interconnect α and β increases the T-switch radix decreases. Note that we are assuming that all the ports provide the same bandwidth (otherwise instead of using the number of ports, the aggregate bandwidth should be used). Consequently, a trade-off between both aspects must be found.

In summary, internal configuration of T-switches, and in general C-switches, becomes a key aspect in the design of this kind of high-radix switches. In the following Section, we present a general methodology to obtain the best configuration of this kind of switches when they are used in large interconnection networks. It is obvious that optimal configuration of a C-switch depends on the conditions under which it is used, that is, network type, topology, routing algorithm, traffic pattern, etc. To show how the methodology works, we apply it to a particular subclass of interconnection network.

2.2 Combined Switches Configuration Methodology

Our methodology to determine the optimal switch-level connection pattern of C-switches consists of the following main steps:

1. Network paths analysis. The purpose of this step is to determine the connections required in each C-switch at network level and the amount of times all these connections are used taking into account all the possible paths used by the packets.

2. Switch classification. Depending on the network characteristics and load conditions, few or many different C-switch configurations could be obtained. In this step, C-switches are grouped according to their connection requirements, and so, several types of C-switch will be distinguished.

   As result of the previous phase, it can occur that some of the possible connections in the C-switches support one, or more paths, and however there may be connections that are never established.

   In a fat-tree topology, for instance, C-switches in different stages may require different switch-level connection patterns, and the same may even occur with C-switches in the same stage. When a simple traffic pattern and balanced routing algorithm are used, it is likely all the C-switches in the network require the same switch-level connection pattern.

3. Switch configuration. From connection requirements and given the number of internal switches forming the C-switch, this last step consists in finding the optimal configuration for each class of C-switch. That is, we must find the optimal switch-level connection pattern of each class, trying to minimize the use of the interconnection between internal switches.

2.3 Study Conditions

As above mentioned, switch-based interconnection networks cover a wide range of network configurations. To present our formal analysis of the C-switches behavior and how they can be configured in an optimal way, we have chosen a very representative case. On the one hand, we consider T-switches due to the reasons outlined in Section 2.1. On the other hand, and since fat-trees are one of the most common topologies today in the largest supercomputers on the Top500 list, we focus on bidirectional interconnection networks (BMINs).

3 Notation

We have assumed the following notation throughout this paper:

- N is the total number of terminals.
- k is the switch arity, or number of ports that connect to terminals/switches in the previous stage and switches in the next stage (if available). Hence, the total number of ports of a $k \times k$ switch is $2k$. The ports faced to the previous stage are numbered from 0 to $k-1$, and the ports connected with the switches in the next stage are labeled from k to $2k-1$.
- Every switch port has an associated global identifier inside the stage, $L = l_{n-1} \ldots l_0, 0 \leq l_i < n$, apart from the internal identifier inside the switch. Both identifiers are related by the connection pattern between stages.

- n is the total number of stages, where $n = \log_k N$.
- h is the terminal identifier ($0 \leq h < N$). It consists of a string of n digits $h_{n-1} \ldots h_1 h_0$ ($0 \leq h_i < k$). $\mathcal{H}$ is the set whose members are the terminals of the MIN, verifying $card(\mathcal{H}) = N$.
- $\langle s, o \rangle$ is a tuple that identifies uniquely a switch, where s refers to the stage ($0 \leq s < n$), and $o = o_{n-2}, \ldots, o_1, o_0$ indicates the position of the switch inside the stage, where $0 \leq o_i < k$ and $0 \leq i < n - 1$.

4 Twin switches

We fully characterize the T-switches by means of the following definitions and propositions.

Definition 1.3 *Let $\mathcal{U}$ be the set of ports on a $k \times k$ switch. Hence,*

$$\mathcal{U} = \{i \in \mathbb{N},\ 0 \leq i < 2k\}$$

Definition 1.4 *Let $\mathcal{B}$ be the set of ports on a $k \times k$ switch connecting to switches from the previous stage (or the input ports in a MIN). Hence,*

$$\mathcal{B} = \{i \in \mathcal{U},\ 0 \leq i < k\}$$

Definition 1.5 *Let $\mathcal{F}$ be the set of ports on a $k \times k$ switch that connect to switches on the next stage in a MIN network. Hence,*

$$\mathcal{F} = \{i \in \mathcal{U},\ k \leq i < 2k\}$$

Figure 2 shows the detailed organization of T-switches. From Definitions 1.3, 1.4, and 1.5 it is obvious that $card(\mathcal{U}) = 2k$ and $card(\mathcal{B}) = card(\mathcal{F}) = k$, where $card$ is the cardinality of sets.

Most of the internal switch ports are dedicated for external communications, meanwhile a concrete number of them are responsible for intra communications between internal switches.

$$\mathcal{B} = \mathcal{B}^{\alpha} \cup \mathcal{B}^{\beta},\ \mathcal{B}^{\alpha} \cap \mathcal{B}^{\beta} = \emptyset$$

$$\mathcal{F} = \mathcal{F}^{\alpha} \cup \mathcal{F}^{\beta},\ \mathcal{F}^{\alpha} \cap \mathcal{F}^{\beta} = \emptyset$$

According to the above, T-switches can be re-defined as follows:

Definition 1.6 *A $k \times k$ T-switch is a bidirectional switch formed by two identical smaller interconnected switches. the $2k$ ports being offered by this T-switch are obtained from the k free ports of each internal switch after they are interconnected by r ports.*

Definition 1.7 *Let $\mathcal{P}^i$ be the set of ports on the switch i, where $i \in \{\alpha, \beta\}$. Moreover, $\mathcal{P}^i$ is divided into three disjoint subsets $\mathcal{J}^i$ to interconnect the internal switches, $\mathcal{B}^i$, and $\mathcal{F}^i$ for the T-switch ports.*

In a more formal way:

- $\mathcal{P}^i = \mathcal{J}^i \cup \mathcal{B}^i \cup \mathcal{F}^i$
- $\mathcal{J}^i \neq \emptyset$
- $\mathcal{B}^i \cap \mathcal{F}^i = \emptyset$, $\mathcal{B}^i \cap \mathcal{J}^i = \emptyset$ *and* $\mathcal{F}^i \cap \mathcal{J}^i = \emptyset$

According to this, it is clear to derive that:

- $\text{card}(\mathcal{J}^\alpha) = \text{card}(\mathcal{J}^\beta)$
- $\text{card}(\mathcal{P}^\alpha) = \text{card}(\mathcal{P}^\beta)$

Proposition 1.1 *In a T-switch consisting of two internal switches α and β, it is verified that*

$$card(\mathcal{B}^\alpha) = card(\mathcal{F}^\beta)$$
$$andcard(\mathcal{B}^\beta) = card(\mathcal{F}^\alpha).$$

Proof: According to Definition 1.7 we know how the switch ports of α and β are configured:

$$\begin{aligned} \mathcal{B} &= \mathcal{B}^\alpha \cup \mathcal{B}^\beta \quad \text{and} \quad \mathcal{B}^\alpha \cap \mathcal{B}^\beta = \emptyset \\ \mathcal{F} &= \mathcal{F}^\alpha \cup \mathcal{F}^\beta \quad \text{and} \quad \mathcal{F}^\alpha \cap \mathcal{F}^\beta = \emptyset \end{aligned}$$

since the sets are disjoint, the expression can be rewritten as

$$\begin{aligned} card(\mathcal{B}) &= card(\mathcal{B}^\alpha) + card(\mathcal{B}^\beta) = k \\ card(\mathcal{F}) &= card(\mathcal{F}^\alpha) + card(\mathcal{F}^\beta) = k \end{aligned}$$

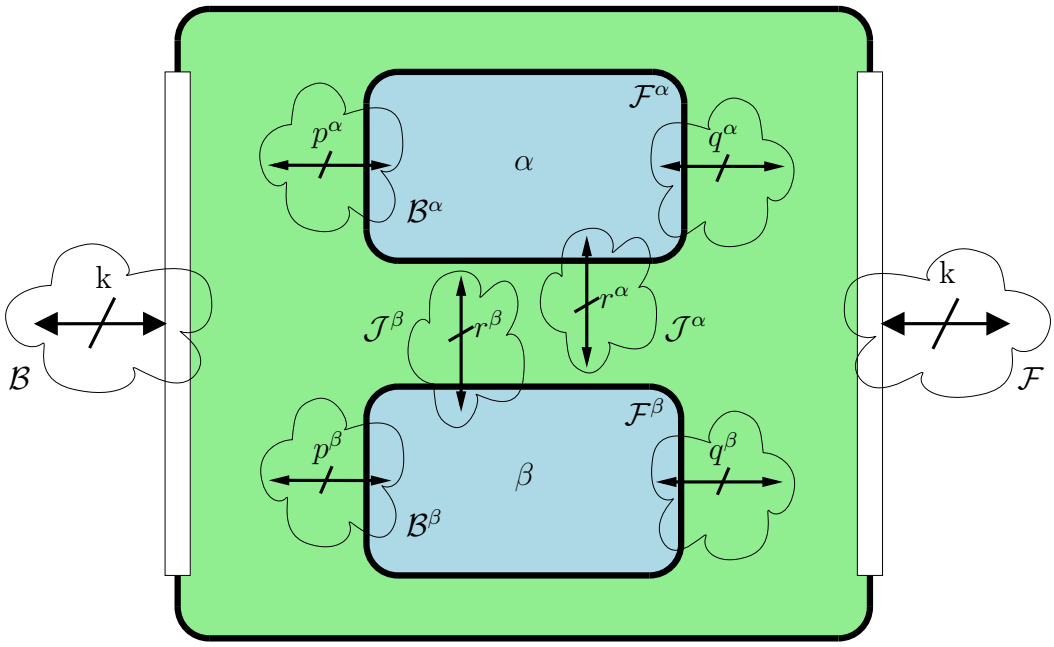


Figure 2: T-switch.

clearing equations

$$card(\mathcal{B}^\beta) = k - card(\mathcal{B}^\alpha) \quad (1)$$
$$card(\mathcal{F}^\alpha) = k - card(\mathcal{F}^\beta) \quad (2)$$

and also

$$card(\mathcal{B}^\alpha) = k - card(\mathcal{B}^\beta) \quad (3)$$
$$card(\mathcal{F}^\beta) = k - card(\mathcal{F}^\alpha) \quad (4)$$

On the other hand, the Definition 1.7 describes how the switch ports of α and β are distributed,

$$\mathcal{P}^\alpha = \mathcal{J}^\alpha \cup \mathcal{B}^\alpha \cup \mathcal{F}^\alpha$$
$$\mathcal{P}^\beta = \mathcal{J}^\beta \cup \mathcal{B}^\beta \cup \mathcal{F}^\beta$$

and also

$$\mathcal{B}^\alpha \cap \mathcal{F}^\alpha = \emptyset,\ \mathcal{B}^\alpha \cap \mathcal{J}^\alpha = \emptyset, \text{ y } \mathcal{F}^\alpha \cap \mathcal{J}^\alpha = \emptyset$$
$$\mathcal{B}^\beta \cap \mathcal{F}^\beta = \emptyset,\ \mathcal{B}^\beta \cap \mathcal{J}^\beta = \emptyset, \text{ y } \mathcal{F}^\beta \cap \mathcal{J}^\beta = \emptyset$$

since $card(\mathcal{P}^\alpha) = card(\mathcal{P}^\beta)$, it is verified that

$$card(\mathcal{J}^\alpha) + card(\mathcal{B}^\alpha) + card(\mathcal{F}^\alpha) = card(\mathcal{J}^\beta) + card(\mathcal{B}^\beta) + card(\mathcal{F}^\beta)$$

considering $card(\mathcal{J}^\alpha) = card(\mathcal{J}^\beta)$, and the expressions 1 and 2, we know that

$$card(\mathcal{B}^\alpha) + k - card(\mathcal{F}^\beta) = k - card(\mathcal{B}^\alpha) + card(\mathcal{F}^\beta)$$
$$2 * card(\mathcal{B}^\alpha) = 2 * card(\mathcal{F}^\beta)$$
$$card(\mathcal{B}^\alpha) = card(\mathcal{F}^\beta)$$

this demonstrates the first clause of the Proposition.

In a similar procedure, but using the equations 3 and 4, the second clause of the Proposition is proved, that is,

$$card(\mathcal{B}^{\beta}) = card(\mathcal{F}^{\alpha})$$

□

As it is suggested by the Figure 2, we assume that:

- $\text{card}(\mathcal{B}^{\alpha}) = p^{\alpha}$.
- $\text{card}(\mathcal{F}^{\alpha}) = q^{\alpha}$.
- $\text{card}(\mathcal{B}^{\beta}) = p^{\beta}$.
- $\text{card}(\mathcal{F}^{\beta}) = q^{\beta}$.

thus from Proposition 1.1, it is possible to conclude that $p^{\alpha} = q^{\beta}$ and $q^{\alpha} = p^{\beta}$.

Accordingly, and in order to use a simpler notation and without loss of accuracy in the final solution, or rigor in the procedure, the following simplifications are assumed:

$$card(\mathcal{B}^{\alpha}) = card(\mathcal{F}^{\beta}) = p. \tag{5}$$
$$card(\mathcal{B}^{\beta}) = card(\mathcal{F}^{\alpha}) = q. \tag{6}$$

Definition 1.8 *Let $\mathcal{C}^i \subset \mathcal{U}$ be the configuration of an internal switch i of a T-switch, where i defines the set of ports on the switch i, which are a subset of the ports of the Tswitch, where $i \in \{\alpha, \beta\}$. Hence,*

$$\mathcal{C}^i = \mathcal{B}^i \cup \mathcal{F}^i$$

Proposition 1.2 *The number of elements in $\mathcal{C}^i$ is k, where $i \in \{\alpha, \beta\}$. In other words $card(\mathcal{C}^i) = k$.*

Proof: The cardinal of $\mathcal{C}^{\alpha}$ is

$$card(\mathcal{C}^{\alpha}) = card(\mathcal{B}^{\alpha}) + card(\mathcal{F}^{\alpha})$$

From the Proposition 1.1, we know that $\text{card}(\mathcal{B}^{\beta}) = \text{card}(\mathcal{F}^{\alpha})$. Thus

$$card(\mathcal{C}^{\alpha}) = card(\mathcal{B}^{\alpha}) + card(\mathcal{B}^{\beta})$$

As defined above in Definition 1.7, $\mathcal{B}^\alpha$ and $\mathcal{B}^\beta$ are disjoint sets and $\mathcal{B} = \mathcal{B}^\alpha \cup \mathcal{B}^\beta$. Therefore, $card(\mathcal{B}) = card(\mathcal{B}^\alpha) + card(\mathcal{B}^\beta)$. By substituting in the above equation, we get

$$card(\mathcal{C}^\alpha) = card(\mathcal{B}) = k$$

Starting from $\mathcal{C}^\beta$ and applying the same process, we will also reach the conclusion that $card(\mathcal{C}^\beta) = k$. □

Definition 1.9 *Let $\mathcal{V}$ be the set of all possible configurations of an internal switch i in a T-switch, where $i \in \{\alpha, \beta\}$. Hence,*

$$\mathcal{V} = \{\mathcal{C}^i \subset \mathcal{U} \mid card(\mathcal{C}^i) = k\}$$

It is important to note that $\mathcal{U}$ is a set, while $\mathcal{V}$ is a set whose members are sets. That is, any configuration $\mathcal{C}^i$ is contained in $\mathcal{U}$, and belongs to $\mathcal{V}$. Formally:

$$\mathcal{C}^i \subset \mathcal{U}$$
$$\mathcal{C}^i \in \mathcal{V}$$

Hereafter, the configuration of an internal switch will be denoted by $\mathcal{C}$, when it does not matter if the switch configuration refers to the switches α or β. The superscript will be only used when necessary to distinguish between α and β.

Definition 1.10 *Let $\mathcal{T}$ be the configuration of T-switch, which is determined by a pair of configurations belonging to $\mathcal{V}$. Hence,*

$$\mathcal{T} = \{\mathcal{C}^\alpha,\ \mathcal{C}^\beta \in \mathcal{V} \mid \mathcal{C}^\beta = (\mathcal{C}^\alpha)^C\}$$

Definition 1.11 *Let γ be a pattern of connections that are applied to a T-switch. Then we define $\mathcal{S}^\gamma$ as the subset of $\mathcal{V}$, whose configurations minimize the number of connections using the ports $\mathcal{J}$ of the internal switches. In other words:*

$$\mathcal{S}^\gamma = \{\mathcal{C} \in \mathcal{V} \ \mid \mathcal{C} \text{ minimizes the use of ports} \mathcal{J}\}$$

Proposition 1.3 *Let $\mathcal{T}$ be a configuration of a T-switch. The configuration $\mathcal{T}$ is an optimal configuration for a specific pattern of connections γ if the configuration of its internal switches belongs to $\mathcal{S}^\gamma$. In other words:*

$$\text{If } \mathcal{T} = \{\mathcal{C}^\alpha,\ \mathcal{C}^\beta \mid \mathcal{C}^\alpha,\ \mathcal{C}^\beta \in \mathcal{S}^\gamma\},\ \text{then } \mathcal{T} \text{ is optimal}$$

Proof: The proof is trivial. The configurations belonging to $\mathcal{S}^{\gamma}$ minimize the number of connections that use the internal link. Consequently, $\mathcal{T}$ is optimal since there are other configurations that minimize the total connections. □

The T-MINs are MIN networks built using T-switches. These networks can be both unidirectional and bidirectional. Our study in this report focuses only on the last ones, however the study of the unidirectional are quite similar.

Definition 1.12 *A T-MIN interconnection network is a MIN network in which all the switches are T-switches.*

Definition 1.13 *A T-BMIN interconnection network is a BMIN network in which all the switches are T-switches.*

4.1 Internal connections of Twin switches

A $k \times k$ T-switch, like any other same size full-duplex switch allows a set of connections between their ports. In particular, we have

- Forward and backward connections. There are $k \times k$ possible combinations that imply pass through the T-switch (in forward and backward directions). We denote by $CC(\langle s, o \rangle)$ (cross connections) the number of different connections between k input ports and k output ports, so $CC(\langle s, o \rangle) = k^2$. If necessary to make a difference between forward and backward directions we will denote by $CC_f(\langle s, o \rangle)$ and $CC_b(\langle s, o \rangle)$, respectively. Hence,

$$CC_f(\langle s, o \rangle) = CC_b(\langle s, o \rangle) = k^2$$

- Turnaround connections. Only k ports take part to establish this type of connection. We denote by $TC(\langle s, o \rangle)$ (turnaround connections) the number of different connections between such ports. It is assumed that there is no turnaround connection between a port and itself. Hence,

$$TC(\langle s, o \rangle) = k(k-1)$$

For the two connection types, considering the internal organization of the T-switch, some of them imply to go across the internal links $\mathcal{J}$ that interconnect the switches α and β. We denote by $CC_I(\langle s, o \rangle)$ the number of paths that go across the $\langle s, o \rangle$ switch by using the switches α and β. Similarly, We denote by $TC_I(\langle s, o \rangle)$ the number of paths that turn around the $\langle s, o \rangle$ switch by using the switches α and β. When necessary, we will differentiate the forward from backward direction by $CC_{If}(\langle s, o \rangle)$ and

$CC_{Ib}(\langle s,o\rangle)$, respectively. In Figure 2, and taking into account the expressions 5 and 6 it is deduced that:

$$CC_I(\langle s,o\rangle) = p{\times}p + q{\times}q = p^2 + (k-p)^2 = 2p^2 + k^2 - 2kp$$
$$TC_I(\langle s,o\rangle) = p{\times}q + q{\times}p = 2p(k-p) = 2(kp - p^2)$$

and also

$$CC_{If}(\langle s,o\rangle) = p{\times}p + q{\times}q = p^2 + (k-p)^2 = 2p^2 + k^2 - 2kp$$
$$CC_{Ib}(\langle s,o\rangle) = p{\times}p + q{\times}q = p^2 + (k-p)^2 = 2p^2 + k^2 - 2kp$$

The total number of times that the internal links connecting the internal switches α and β on a $\langle s,o\rangle$ T-switch would be used, is denoted by $C_i(\langle s,o\rangle)$, and it is obtained from the $CC_I(\langle s,o\rangle)$ ($CC_{If}(\langle s,o\rangle)$ and $CC_{Ib}(\langle s,o\rangle)$ if applicable) and $TC_I(\langle s,o\rangle)$ in each case are obtained under the initial conditions.

In the previous, we saw the expressions $CC_I(\langle s,o\rangle)$ and $TC_I(\langle s,o\rangle)$ that have been obtained considering the isolated T-switch, without taking into account the entire network; nor the paths that are routed by the concrete routing algorithm; nor the characteristics of the traffic that determine the paths. Once all these aspects are considered, it is possible to determine what happens in individual cases and therefore we can get the configuration of all network T-switches to minimize the number of crosses by the internal links [2] that interconnect the switches α and β.

In such cases, the number of paths that pass through switches can be calculated using these expressions or not (depending on the characteristics of each case), all the paths that pass through or turn on each $\langle s,o\rangle$ switch, and which of them make it through the switches α and β.

To obtain these expressions we need to know when a path reaches the switch $\langle s,o\rangle$. Considering that is a BMIN network topology of N nodes and n stages, a switch $\langle s,o\rangle$ will be achieved:

- from a switch located in a previous stage or from one terminal node to reach a later stage. In this case the path goes across the switch. We denote by $C_f(\langle s,o\rangle)$ (a.k.a. forward crosses) the number of paths that go across the switch $\langle s,o\rangle$, $0 \leq s < n-1$, in the forward direction. This is true for switches belonging to all stages except the last.
- from a switch located in a later stage to arrive at an previous stage or a terminal node. In this case the path goes across the switch. We denote by $C_b(\langle s,o\rangle)$ (a.k.a. backward crosses) the number of paths that go across the switch $\langle s,o\rangle$,

[2] In what follows, we will also use the term "internal link" to refer to the ports $\mathcal{J}$ that connect the switches α and β.

$0 \leq s < n-1$, in the backward direction. This is also true for switches belonging to all stages except the last.

- from a switch located in a previous stage or a terminal node to reach another different switch in the same stage or different terminal node. In this case there is a twist on the switch itself. We denote by $T(\langle s, o\rangle)$ (a.k.a. turnaround connection) the number of paths that turn around the switch $\langle s, o\rangle$. Unlike earlier, this is true in all switches.

Sometimes it will be convenient to consider $C_f(\langle s, o\rangle)$ and $C_b(\langle s, o\rangle)$ together. Thus, we also introduce $C(\langle s, o\rangle)$ (total crosses) as

$$C(\langle s, o\rangle) = C_f(\langle s, o\rangle) + C_b(\langle s, o\rangle)$$

The above expressions are switch-level expressions, but they do not distinguish between individual ports. However, for this study it is necessary to know which connections are established between individual ports and how many times. This information will determine which of them may be made without using the internal switches α and β.

Therefore, we will also consider counters similar to those introduced above, but at connection level. But we will distinguish among those going in the forward, downward, or turn-around connections. To register the number of occurrences of each event will be used, $C_f(\langle s, o\rangle, l, l')$, $C_b(\langle s, o\rangle, l, l')$ and $T(\langle s, o\rangle, l, l')$, respectively. In short:

- $C_f(\langle s, o\rangle, l, l')$ records the number of times it is used the connection between the ports l and l', with $0 \leq l < k$ and $k \leq l' < 2k$.
- $C_b(\langle s, o\rangle, l, l')$ records the number of times it is used the connection between the ports l and l', with $k \leq l < 2k$ and $0 \leq l' < k$.
- $T(\langle s, o\rangle, l, l')$ records the number of times it is used the connection between the ports l and l', with $0 \leq l, l' < k$ and $l \neq l'$. The sum of the first two are denoted by $C(\langle s, o\rangle, l, l')$.

Adding the first two we obtain $C(\langle s, o\rangle, l, l')$.

5 Applying the Methodology

Notice that although the methodology is general and it can be applied considering different C-switches and interconnection networks, the optimal T-switch configuration depends on the particular network properties. In this case we have considered the following network properties: BMINs k–ary n–tree with N terminals and $k \times k$ T-switches ($k \geq 4$), DESTRO routing algorithm, and uniform traffic pattern.

Therefore, in order to apply the switch configuration methodology we must specify the network topology, the routing algorithm, and the network load.

We have chosen the BMIN k–ary n–tree network [DYN03], a subclass of fat-trees which are one of the most common topologies today in the largest supercomputers on the Top500 list. The k–ary n–tree network topology belongs to the family of fat–trees and it is derived from a concrete class of MINs: the k–ary n–butterflies (or k–ary n-flies) [Lei92]. A k–ary n–fly MIN is obtained by applying the β_i^k permutation, $0 \leq i < n$, to obtain the network-level connection patterns between stages. The k–ary n–tree connect N nodes using nk^{n-1} switches. Two switches $\langle s, o_{n-2} \dots o_0 \rangle$ and $\langle s', o'_{n-2} \dots o'_0 \rangle$ are connected with a link if $s' = s + 1$ and $o_i = o'_i \ \forall i \neq s$. Moreover, there is a link between the switch $\langle 0, o_{n-2} \dots o_0 \rangle$ and the terminal $h = h_{n-1} \dots h_0$ if $o_i = h_{i+1}, 0 \leq i < n - 1$.

The routing algorithm is DESTRO [GGG+07]. It is a deterministic routing algorithm for fat-trees. It is based on using at each switch the ascending output port given by the destination component of the packet that is being routed corresponding to the switch stage. This routing algorithm is able to evenly balance network traffic and reduce to the minimum the number of paths that share each link, and as a consequence, it reduces network contention.

Uniform traffic pattern is assumed as network workload because it is a common pattern in many studies about interconnection networks.

Under these conditions, we formally demonstrate what is the best configuration of the T-switches. The following three sections correspond to the steps in the methodology.

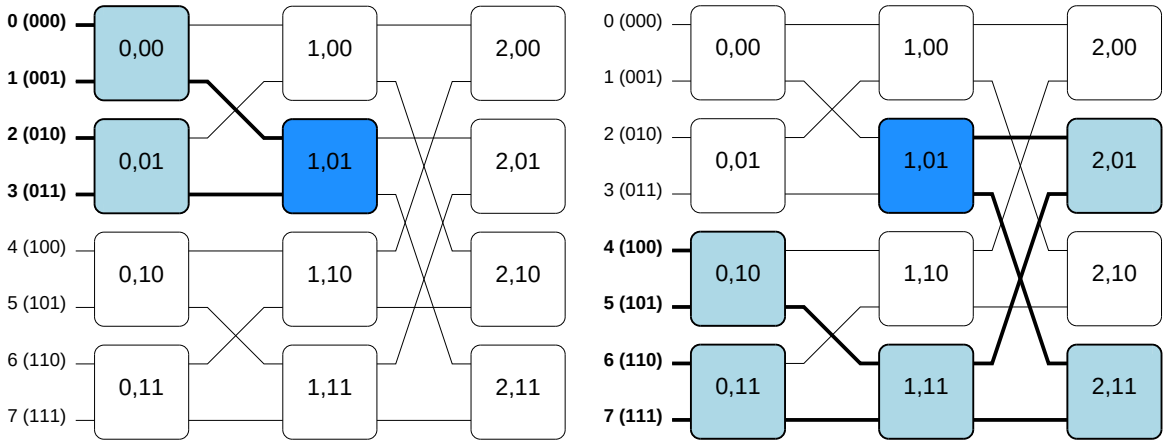
5.1 Networks paths analysis

For a further treatment, it is interesting to know the reachable nodes from a particular BMIN switch. We have distinguished two cases: (1) in the first case we only take into account the topological capabilities of butterfly BMINs; (2) however in the second case, we have additionally considered a routing algorithm. In both cases, we introduce some definitions and derived propositions. Before every definition, we give an example for a sake of clarity.

In all the examples, we assume a 2-ary 3-tree BMIN with $N = 8$ nodes, and consider the $\langle 1, 01 \rangle$ (dark blue) as the reference switch. On the other hand, we highlight in light blue the switches in the middle of the reference switch and the reachable nodes.

5.1.1 Reachable nodes from a BMIN switch considering the network topology

Example 1.1 *Topologically speaking, a path can reach the nodes {0,1,2,3} in the descending phase, and the nodes {4,5,6,7} in the ascending phase, from the* $\langle 1, 01 \rangle$ *switch, as it can be seen in the Figures 3(a) and 3(b), respectively.*



(a) Descending phase. (b) Ascending phase.

Figure 3: Reachable nodes from a switch considering the topology.

Example 1.2 *Topologically speaking, a path can reach the nodes {0,1} in the descending phase by using the output port 0, and the nodes {4,5,6,7} in the ascending phase by using the output port 2, from the* $\langle 1, 01 \rangle$ *switch, as it can be seen in the figures 4(a) and 4(b), respectively.*

In a more formal way, given a k-ary n-tree BMIN network with N nodes, we introduce the following definitions:

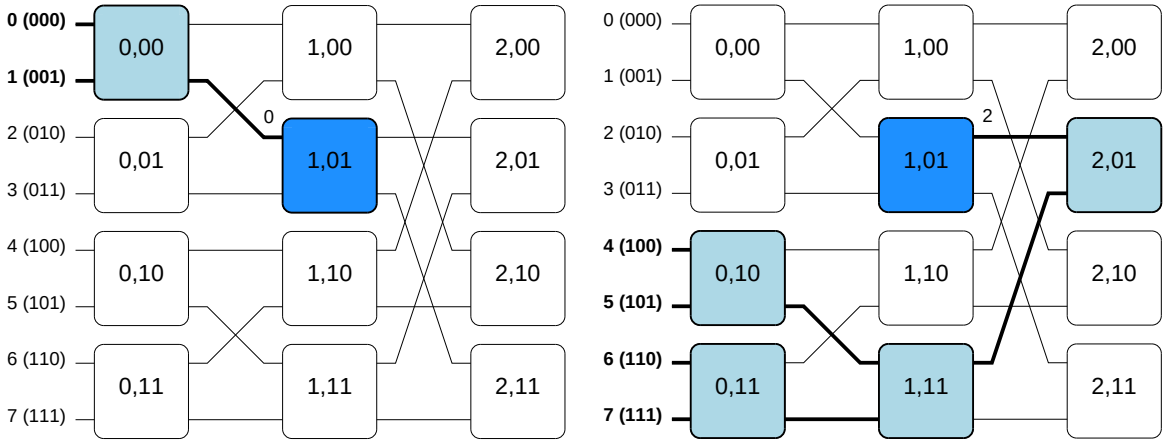
Definition 1.14 *Let* $N_b^t(\langle s, o \rangle)$ *be the network node set that are topologically reachable from the switch* $\langle s, o \rangle$ *by a path in the descending phase, where* $\langle s, o \rangle = \langle s, (o_{n-2} \dots o_0) \rangle$ *for* $0 \leq s < n$*. Hence*

$$N_b^t(\langle s, o \rangle) = \{\ (h_{n-1} \dots h_0) :\ h_i = o_{i-1}\ \forall i \in [s+1, n-1]\ \}$$

Definition 1.15 *Let* $N_f^t(\langle s, o \rangle)$ *be the network node set that are topologically reachable from the switch* $\langle s, o \rangle$ *by a path in the ascending phase, where* $\langle s, o \rangle = \langle s, (o_{n-2} \dots o_0) \rangle$ *for* $0 \leq s < n-1$*. Hence*

$$N_f^t(\langle s, o \rangle) = (N_b^t(\langle s, o \rangle))^C = \{\ (h_{n-1} \dots h_0) :\ \exists i \in [s+1, n-1] \mid h_i \neq o_{i-1}\ \}$$

where C *refers to set complement operation.*



(a) Descending phase by the output port 0. (b) Ascending phase by the output port 2.

Figure 4: Reachable nodes through a port from a switch considering the topology.

Similarly, we also define the reachable node set by the output port l in the switch $\langle s, o\rangle$.

Definition 1.16 *Let $N_b^t(\langle s, o\rangle, l)$ be the network node set that are topologically reachable by the output port l from the switch $\langle s, o\rangle$ by a path in the descending phase, where $\langle s, o\rangle = \langle s, (o_{n-2} \dots o_0)\rangle$ for $0 \leq s < n$ and $0 \leq l < k$. Hence*

$$N_b^t(\langle s, o\rangle, l) = \{ (h_{n-1} \dots h_0) : h_i = o_{i-1} \ \forall i \in [s+1, n-1] \ y \ h_s = l \}$$

Definition 1.17 *Let $N_f^t(\langle s, o\rangle, l)$ be the network node set that are topologically reachable by using the output port l from the switch $\langle s, o\rangle$ by a path in the ascending phase, where $\langle s, o\rangle = \langle s, (o_{n-2} \dots o_0)\rangle$ for $0 \leq s < n-1$ and $k \leq l < 2k$. Hence*

$$N_f^t(\langle s, o\rangle, l) = \{ (h_{n-1} \dots h_0) : \exists i \in [s+1, n-1] \mid h_i \neq o_{i-1} \}$$

Taking the previous definitions as the starting point, the following propositions can be considered:

Proposition 1.4 *The total number of topologically reachable network nodes from the switch $\langle s, o\rangle$ by a path in the descending phase is $D_b^t(\langle s, o\rangle) = k^{s+1}$.*

Proof: Every node identifier can be written as a sequence of n digits:

$$(h_{n-1} \dots h_{s+1} h_s h_{s-1} \dots h_0)$$

and according to Definition 1.14, the sequence can be rewritten as:

$$(o_{n-2} \dots o_{s+1} o_s h_s h_{s-1} \dots h_0)$$

that is, the $n-1-s$ most significant digits $o_{n-2} \dots o_s$ are the same for all the set members. The $s+1$ remaining digits $h_s \dots h_0$ may be different and distinguish the set nodes between them. Every h_i would take values inside the range [0, k], that is, k different values. Therefore, the number of different sequences, or the number of set nodes is k^{s+1}. □

Proposition 1.5 *The total number of topologically reachable network nodes from the switch $\langle s, o\rangle$ by a path in the ascending phase is $D_f^t(\langle s, o\rangle) = k^n - k^{s+1}$.*

Proof: According to Definition 1.15

$$N_f^t(\langle s, o\rangle) = (N_b^t(\langle s, o\rangle))^C$$

therefore

$$\begin{aligned} D_f^t(\langle s, o\rangle) &= card(N_f^t(\langle s, o\rangle)) = card((N_b^t(\langle s, o\rangle))^C) = \\ &= card(\mathcal{H}) - card(N_b^t(\langle s, o\rangle)) = N - card(N_b^t(\langle s, o\rangle)) = N - k^{s+1} = k^n - k^{s+1} \end{aligned}$$

□

Proposition 1.6 *The total number of topologically reachable network nodes by using the output port l from the switch $\langle s, o\rangle$ by a path in the descending phase, where $0 \le s < n$ and $0 \le l < k$, is $D_b^t(\langle s, o\rangle, l) = k^s$.*

Proof: If $s > 0$, the topologically reachable network nodes by using the output port l from the switch $\langle s, o\rangle$ by a path in the descending phase are the same as the reachable nodes from the switch $\langle s-1, o'\rangle$, which $\langle s, o\rangle$ is connected to through its port l, hence

$$D_b^t(\langle s, o\rangle, l) = D_b^t(\langle s-1, o'\rangle) = k^{(s-1)+1} = k^s$$

If $s = 0$, only one node is connected through the port l, and $D_b^t(\langle 0, o\rangle, l) = k^s$ is also verified because $k^0 = 1$. □

Proposition 1.7 *The total number of topologically reachable network nodes by using the output port l from the switch $\langle s, o\rangle$ by a path in the ascending phase, where $0 \le s < n-1$ and $k \le l < 2k$, is $D_f^t(\langle s, o\rangle, l) = k^n - k^{s+1}$.*

Proof: According to Definitions 1.15 and 1.17, the sets $N_f^t(\langle s, o\rangle)$ and $N_f^t(\langle s, o\rangle, l)$ have the same members. Hence,

$$D_f^t(\langle s, o\rangle, l) = D_f^t(\langle s, o\rangle) = k^n - k^{s+1}$$

□

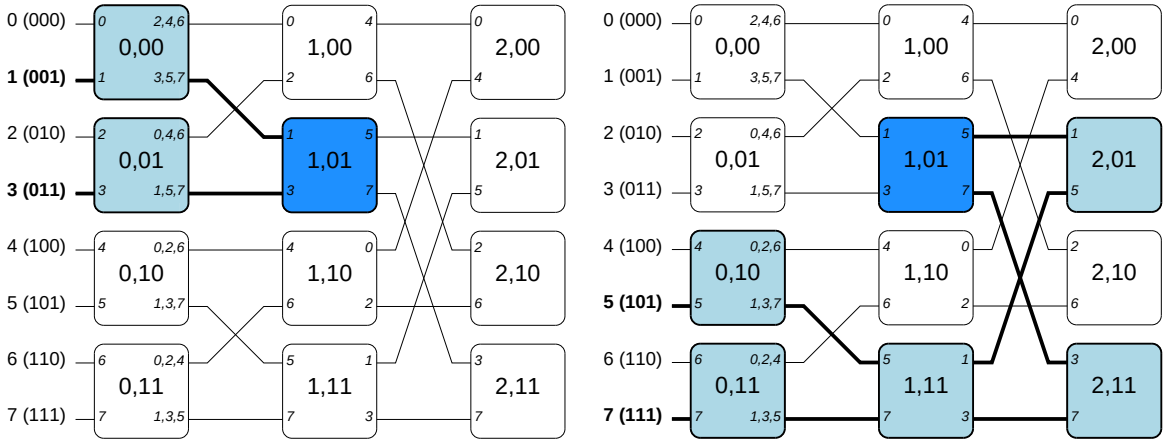
5.1.2 Reachable nodes from a BMIN switch considering the network topology and the routing algorithm

Similar to Section 5.1.1, we also give some examples to understand the following definitions and propositions. The reference switch and the reachable nodes are highlighted in dark and light blue, respectively. Furthermore, every output port is labeled with the reachable network nodes identifier, which are calculated by a routing algorithm likewise the described algorithm in Section A.7.1.

Considering the routing algorithm, we have to take into account that given a switch, i.e. $\langle 1, 01 \rangle$, only a concrete destinations are reachable:

Example 1.3 *From switch $\langle 1, 01 \rangle$, the reachable destinations are {1,3} in the descending phase, and {5,7} in the ascending phase, as it can be seen in the figures 5(a) and 5(b), respectively.*

For the reason stated just before starting this example, nodes 0 and 2 are not reachable because the routing algorithm prevents (in every hop) paths for reaching the switch $\langle 1, 01 \rangle$.

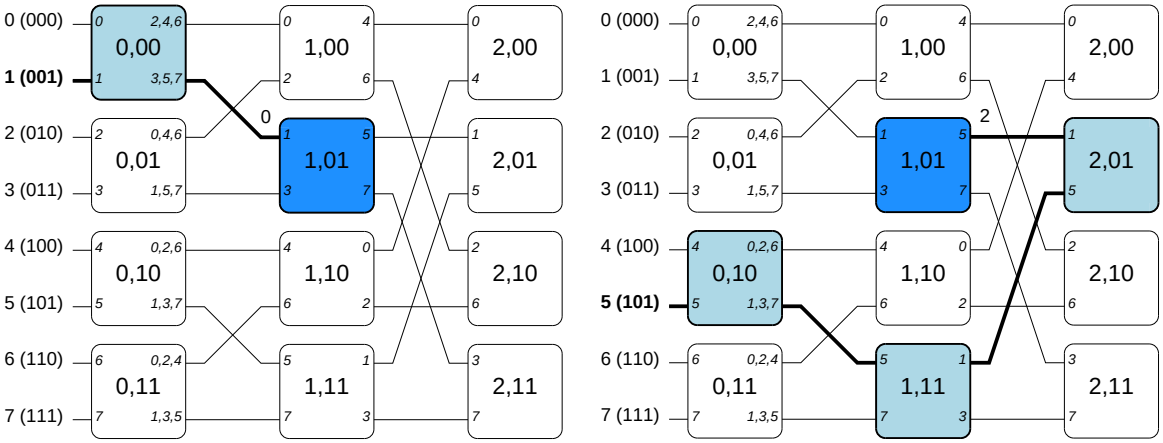


(a) Descending phase. (b) Ascending phase.

Figure 5: Reachable nodes from the switch considering the routing algorithm.

Example 1.4 *Let us take the switch $\langle 1, 01 \rangle$ as the reference point. Through the port 0, a path can only reach the node {1} in the descending phase; but through the port 2, the path can only reach the node {5} in the ascending phase. This can be observed in the figures 6(a) and 6(b), respectively.*

In a more formal way, given a k-ary n-tree BMIN network with N nodes and considering **DESTRO** (Section A.7.1) as the routing algorithm, R, we introduce the definitions below:



(a) Descending phase by the output port 0.

(b) Sentido ascendente by the output port 2.

Figure 6: Reachable nodes through a port from a switch considering the routing algorithm.

Definition 1.18 *Let $N_b^R(\langle s, o\rangle)$ be the reachable network nodes from the switch $\langle s, o\rangle$ by a path in the descending phase, considering R, where $\langle s, o\rangle = \langle s, (o_{n-2} \dots o_0)\rangle$, $0 \leq s < n$. Hence*

$$N_b^R(\langle s, o\rangle) = \{\ (h_{n-1} \dots h_0) :\ h_i = o_{i-1}\ \forall i \in [s+1, n-1],\ h_i = o_i\ \forall i \in [0, s-1]\ \}$$

Definition 1.19 *Let $N_f^R(\langle s, o\rangle)$ be the reachable network nodes from the switch $\langle s, o\rangle$ by a path in the ascending phase, considering R, where $\langle s, o\rangle = \langle s, (o_{n-2} \dots o_0)\rangle$, $0 \leq s < n-1$. Hence*

$$N_f^R(\langle s, o\rangle) = \{\ (h_{n-1} \dots h_0) :\ \exists i \in [s{+}1, n{-}1] \mid h_i \neq o_{i-1}\ \textit{and}\ h_i = o_i\ \forall i \in [0, s{-}1]\ \}$$

Similarly, we also define the reachable node set by the output port l in the switch $\langle s, o\rangle$.

Definition 1.20 *Let $N_b^R(\langle s, o\rangle, l)$ be the reachable network node set by using the output port l from the switch $\langle s, o\rangle$ by a path in the descending phase, considering R, where $\langle s, o\rangle = \langle s, (o_{n-2} \dots o_0)\rangle$ with $0 \leq s < n$ and $0 \leq l < k$. Hence*

$$N_b^R(\langle s, o\rangle, l) = \{\ (h_{n-1} \dots h_0) :\ h_i = o_{i-1}\ \forall i \in [s+1, n-1],\ h_s = l,\ \textit{and}\ h_i = o_i\ \forall i \in [0, s-1]\ \}$$

Definition 1.21 *Let $N_f^R(\langle s,o\rangle,l)$ be the reachable network node set by using the output port l from the switch $\langle s,o\rangle$ by a path in the ascending phase, considering R, where $\langle s,o\rangle = \langle s,(o_{n-2}\dots o_0)\rangle$ with $0 \leq s < n-1$ and $k \leq l < 2k$. Hence*

$$N_f^R(\langle s,o\rangle,l) = \{\ (h_{n-1}\dots h_0) :\ \exists i \in [s+1,n-1] \mid h_i \neq o_{i-1},\ and \\ h_s = l-k,\ y\ h_i = o_i\ \forall i \in [0,s-1]\ \}$$

Taking the previous definitions as the starting point, the following propositions can be considered:

Proposition 1.8 *The number of reachable network nodes from the switch $\langle s,o\rangle$ by a path in the descending phase, considering R, is $D_b^R(\langle s,o\rangle) = k$.*

Proof: The network node identifier is a sequence of n digits:

$$(h_{n-1}\dots h_{s+1}h_s h_{s-1}\dots h_0)$$

and according to Definition 1.18, the sequence can be rewritten as:

$$(o_{n-2}\dots o_{s+1}o_s h_s o_{s-1}\dots o_0)$$

All the digits are the same as those of the switch $\langle s,o\rangle$, with the exception of the digit h_s. Therefore, the digit h_s makes the difference of the node identifiers in the set. Since h_s could take k values ($0 \leq h_s < k$), there are k different members belonging to the set $N_b^R(\langle s,o\rangle)$, and consequently $D_b^R(\langle s,o\rangle) = k$. □

Proposition 1.9 *The number of reachable network nodes from the switch $\langle s,o\rangle$ by a path in the ascending phase, considering R, is $D_f^R(\langle s,o\rangle) = k^{n-s} - k$.*

Proof: Every node identifier is a sequence of n digits with the format below:

$$(h_{n-1}\dots h_{s+2}h_{s+1}h_s h_{s-1}\dots h_0)$$

and according to Definition 1.19, such a sequence can be rewritten as:

$$(h_{n-1}\dots h_{s+2}h_{s+1}h_s o_{s-1}\dots o_0)$$

The Definition 1.19 states the $n-s-1$ most significant digits, $h_{n-2}\dots h_s$ provide k^{n-s-1} different combinations of valid sequences. However, there is only one combination that does not verify the Definition 1.19. Specifically, it is that which has

$h_{n-1} \dots h_{s+1} = o_{n-2} \dots o_s$. In that way, the number of combinations for the $n - s - 1$ left most significant digits is $k^{n-s-1} - 1$.

On the other hand, the s least significant digits, $h_{s-1} \dots h_0$, are fixed by Definition 1.19.

Finally, the digit h_s would take one value of k possibilities. Therefore, the number of possible combinations for node identifiers, that is, the members of the set $N_f^R(\langle s, o\rangle)$ is $(k^{n-s-1} - 1) \times k = k^{n-s} - k$. □

Proposition 1.10 *The number of reachable network nodes by using the output port l from the switch $\langle s, o\rangle$ by a path in the descending phase, considering R, where $0 \leq l < k$ and $0 \leq s < n$, is $D_b^R(\langle s, o\rangle, l) = 1$.*

Proof: According to Proposition 1.8, the number of reachable network nodes from the switch $\langle s, o\rangle$ by a path in the descending phase is k, and because R is balanced, those nodes are uniformly distributed between the k output ports, and therefore

$$D_b^R(\langle s, o\rangle, l) = D_b^R(\langle s, o\rangle)/k = k/k = 1$$

□

Proposition 1.11 *The number of reachable network nodes through the output port l from the switch $\langle s, o\rangle$ by a path in the ascending phase, considering R, where $k \leq l < 2k$ and $0 \leq s < n - 1$ is $D_f^R(\langle s, o\rangle, l) = k^{n-s-1} - 1$.*

Proof: According to Proposition 1.9, the number of reachable nodes from the switch $\langle s, o\rangle$ by a path in the ascending phase is $k^{n-s} - k$, and since R is balanced, those nodes are uniformly distributed between the k output ports, and therefore

$$D_f^R(\langle s, o\rangle, l) = D_f^R(\langle s, o\rangle)/k = (k^{n-s} - k)/k = k^{n-s-1} - 1$$

□

Proposition 1.12 *Given an input port l, $k \leq l < 2k$, an output port l', $0 \leq l' < k$, and both belong to a switch $\langle s, o\rangle$, where $\langle s, o\rangle = \langle s, (o_{n-2} \dots o_0)\rangle$ with $0 \leq s < n - 1$, considering R, there exist paths in the descending phase that arrive at $\langle s, o\rangle$ by l and leave it through l' if and only if $l' = l - k$ is verified.*

Proof: Let us suppose that a path in the descending phase arrives at the switch $\langle s, o\rangle$, and let h' be the destination node of that path. The path would leave the switch through the output port l' in the switch $\langle s, o\rangle$.

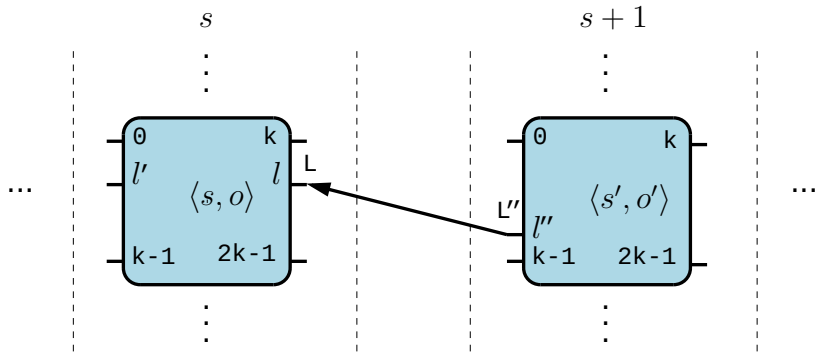


Figure 7: Associated channel between two adjacent switches.

Since R is the routing algorithm, the Definition 1.20 determines the reachable nodes by a path in the descending phase in $\langle s, o\rangle$ by using l'. According to Proposition 1.10, such a set has only one member, this is, the destination node h':

$$N_b^R(\langle s, o\rangle, l') = \{\ h'\ \} = \{\ (o_{n-2} \dots o_{s+1} o_s l' o_{s-1} \dots o_0)\ \}$$

Since the path is in the descending phase, it comes from one of the switches placed in the upwards stage $\langle s', o'\rangle$, with $s' = s + 1$. If a path passes through two switches, each one belonging to different stages, both switches are connected by a channel. For example, in the Figure 7, the switches $\langle s, o\rangle$ and $\langle s', o'\rangle$ are connected through the ports l and l''. Taking into account the k-ary n-tree topology definition, it is known the switch identifiers verify $o_i = o'_i\ \forall i \neq s$, in other words:

$$\langle s', o'\rangle = \langle s + 1, (o_{n-2} \dots o_{s+1} o'_s o_{s-1} \dots o_0)\rangle$$

It should be notice all the values that o' can take are known except o'_s.

On the other hand, the Definition 1.20 specifies the reachable network node set in the descending phase from the switch $\langle s', o'\rangle$ through the output port l''. Such a set has only one member: the destination node h':

$$N_b^R(\langle s', o'\rangle, l'') = \{h'\} = \{\ (o_{n-2} \dots o_{s+1} l'' o'_s o_{s-1} \dots o_0)\ \}$$

Since the sets $N_b^R(\langle s, o\rangle, l')$ and $N_b^R(\langle s', o'\rangle, l'')$ have the same members, $\{h'\}$,

$$\begin{aligned} N_b^R(\langle s, o\rangle, l') &= \{\ (o_{n-2} \dots o_{s+1} o_s l' o_{s-1} \dots o_0)\ \} \\ N_b^R(\langle s', o'\rangle, l'') &= \{\ (o_{n-2} \dots o_{s+1} l'' o'_s o_{s-1} \dots o_0)\ \} \end{aligned}$$

that is checked only if:

$$\begin{aligned} o'_s &= l' \\ o_s &= l'' \end{aligned}$$

Let us analyze the relation between l and l'' with the connection pattern. The pattern was defined in Section A.2 in terms of global ports L and L''. The number of

internal port l, $k \leq l < 2k$, is associated with $L = l_{n-1} \dots l_0$ by the connection pattern as:

$$L = k \times o + (l - k) = k \times (o_{n-2} \dots o_0) + (l - k) = o_{n-2} \dots o_0 (l - k)$$

The multiplication of $o = o_{n-2} \dots o_0$ (in base k) by k is calculated by shifting $o = o_{n-2} \dots o_0$ left one position, and assigning the digit $o_0 = 0$. Then the addition operation sets the digit o_0 to $l - k$ ($0 \leq l - k < k$).

Similarly, the global port number $L'' = l''_{n-1} \dots l''_0$ is related with the internal port l'', $0 \leq l'' < k$, by the expression below:

$$L'' = k \times o' + l'' = k \times (o'_{n-2} \dots o'_0) + l'' = o'_{n-2} \dots o'_0 l''$$

Then again, the butterfly permutation associates the ports L and L'' as follows:

$$\begin{aligned} \beta^k_{s+1}(L) &= L'' \\ \beta^k_{s+1}(o_{n-2} \dots o_{s+1} o_s o_{s-1} \dots o_0 (l - k)) &= o'_{n-2} \dots o'_{s+1} o'_s o'_{s-1} \dots o'_0 l'' \end{aligned}$$

taking into account that $o'_s = l'$ and $l'' = o_s$, it is obtained in L''

$$\beta^k_{s+1}(o_{n-2} \dots o_{s+1} o_s o_{s-1} \dots o_0 (l - k)) = o'_{n-2} \dots o'_{s+1} l' o'_{s-1} \dots o'_0 o_s \qquad (7)$$

how the switches $\langle s, o \rangle$ and $\langle s', o' \rangle$ are related

$$\langle s', o' \rangle = \langle s + 1, (o_{n-2} \dots o_{s+1} o'_s o_{s-1} \dots o_0) \rangle$$

and it can be substituted in the Expression 7, remaining then

$$\beta^k_{s+1}(o_{n-2} \dots o_{s+1} o_s o_{s-1} \dots o_0 (l - k)) = o_{n-2} \dots o_{s+1} l' o_{s-1} \dots o_0 o_s$$

at long last, the butterfly permutation is applied

$$o_{n-2} \dots o_{s+1} (l - k) o_{s-1} \dots o_0 o_s = o_{n-2} \dots o_{s+1} l' o_{s-1} \dots o_0 o_s \qquad (8)$$

The equality 8 is verified when $l' = l - k$. □

It should be remembered that the purpose of this first step is to know the connections required in each T-switch that is to obtain $C_f(\langle s, o \rangle)$, $T(\langle s, o \rangle)$ and $C_b(\langle s, o \rangle)$ both at switch and port level.

Firstly, the generated paths with uniform traffic pattern are studied. This traffic pattern generates $N^2 - N$ different paths between N terminals in the network. Since we are assuming a deterministic routing algorithm, there is a unique path between every pair of terminals. Moreover, as the routing algorithm is load-balanced, the paths in the ascending, turnaround and descending phase are uniformly distributed among all the switches at the same stage.

Then, the number of paths passing through the switch $\langle s, o \rangle$ in the ascending, turnaround and descending phases, $0 \leq s < n$, are determined. In order to get a more clear and simple treatment, we will avoid continually repeating the starting conditions in every definition and proposition.

5.1.3 Ascending phase of the paths

Some propositions related to the paths passing through the switch $\langle s, o\rangle$ in the ascending phase, $0 \leq s < n$, are described below.

Proposition 1.13 *Given the ports l and l' of the switch $\langle s, o\rangle$, $0 \leq s < n-1$, $0 \leq l < k$ and $k \leq l' < 2k$, there are $k^{n-1} - k^s$ paths passing through $\langle s, o\rangle$ in the ascending phase by using l and l'.*

Proof: A path passes through the switch $\langle s, o\rangle$ in the ascending phase by using the ports l and l' if the source node, h, belongs to the set $N_b^t(\langle s, o\rangle, l)$, the destination node, h', belongs to the set $N_f^R(\langle s, o\rangle, l')$. Therefore, the total number of paths that pass through $\langle s, o\rangle$ in the ascending phase by using l and l' is equal to the number of source–destination pairs of nodes obtained from these sets.

$$C_f(\langle s, o\rangle, l, l') = card(N_b^t(\langle s, o\rangle, l)) \times card(N_f^R(\langle s, o\rangle, l')) = \\ = D_b^t(\langle s, o\rangle, l) \times D_f^R(\langle s, o\rangle, l') = k^s \times (k^{n-s-1} - 1) = k^{n-1} - k^s$$

that is

$$C_f(\langle s, o\rangle, l, l') = k^{n-1} - k^s$$

□

Proposition 1.14 *The number of paths in ascending phase passing through the switch $\langle s, o\rangle$, $0 \leq s < n-1$, is $k^{n+1} - k^{s+2}$.*

Proof: The total number of paths in the ascending phase passing through the switch $\langle s, o\rangle$ is given by the multiplication of the number of paths that use l and l' ($0 \leq l < k$ and $k \leq l' < 2k$) by the number of pairs (l,l').

$$C_f(\langle s, o\rangle) = C_f(\langle s, o\rangle, l, l') \times k^2 = (k^{n-1} - k^s) \times k^2 = k^{n+1} - k^{s+2}$$

that is

$$C_f(\langle s, o\rangle) = k^{n+1} - k^{s+2}$$

□

Proposition 1.15 *There is no path passing through, in the ascending phase, the switches of the last stage. Hence,*

$$C_f(\langle n-1, o\rangle) = C_f(\langle n-1, o\rangle, l, l') = 0$$

Proof: By the definition of the BMIN topology, there are no forward connections in the switches of the last stage. □

5.1.4 Turnaround phase of the paths

Some propositions related to the paths passing through the switch $\langle s, o\rangle$ in the turnaround phase, $0 \leq s < n$, are described below.

Proposition 1.16 *Given the ports l and l' of the switch $\langle s, o\rangle$, where $0 \leq s < n$, $0 \leq l, l' < k$ and $l \neq l'$, there are k^s paths passing through $\langle s, o\rangle$ in the turnaround phase by using l and l'.*

Proof: A path is turned around in the switch $\langle s, o\rangle$ by the ports l and l' if the source node, h, belongs to the set $N_b^t(\langle s, o\rangle, l)$ and the destination node, h', belongs to the set $N_b^R(\langle s, o\rangle, l')$. Therefore, the total number of paths passing through the switch $\langle s, o\rangle$ in the turnaround phase by using l and l' is equal to number of source–destination node pairs obtained from these sets.

$$T(\langle s, o\rangle, l, l') = card(N_b^t(\langle s, o\rangle, l)) \times card(N_b^R(\langle s, o\rangle, l')) = \\ = D_b^t(\langle s, o\rangle, l) \times D_b^R(\langle s, o\rangle, l') = k^s \times 1 = k^s$$

that is

$$T(\langle s, o\rangle, l, l') = k^s$$

□

Proposition 1.17 *The number of paths that turn around in the switch $\langle s, o\rangle$, $0 \leq s < n$, is $(k-1)k^{s+1}$.*

Proof: The total number of paths passing through the switch $\langle s, o\rangle$ in the turnaround phase is obtained by the multiplication of the number of paths that use l and l' ($0 \leq l < k$ and $k \leq l' < 2k$) by the number of pairs (l,l'). The first element of the multiplication is obtained by the Proposition 1.16 and the second one, taking into account that no path turns in a switch using the same port as input and output.

$$T(\langle s, o\rangle) = T(\langle s, o\rangle, l, l') \times k(k-1) = k^s \times k(k-1) = (k-1)k^{s+1}$$

that is

$$T(\langle s, o\rangle) = (k-1)k^{s+1}$$

□

5.1.5 Descending phase of the paths

Some propositions related to the paths passing through the switch $\langle s, o\rangle$ in the descending phase, $0 \leq s < n$, are described below.

Proposition 1.18 *There is no path passing through, in the descending phase, the switches of the last stage. Hence,*

$$C_b(\langle n-1, o\rangle) = C_b(\langle n-1, o\rangle, l, l') = 0$$

Proof: By the definition of the BMIN topology, there are no backward connections in the switches of the last stage. □

Proposition 1.19 *Given the ports l and l' of the switch $\langle s, o\rangle$, where $0 \leq s < n-1$, $k \leq l < 2k$ and $0 \leq l' < k$, at the most there exist $k^n - k^{s+1}$ paths passing through $\langle s, o\rangle$ in the descending phase by using l and l'.*

Proof: A path goes through the switch $\langle s, o\rangle$ in the descending phase by using the ports l and l' if the source node, h, belongs to the set $N_f^t(\langle s, o\rangle, l)$ and the destination node, h', belongs to the set $N_b^R(\langle s, o\rangle, l')$. Therefore, the total number of paths passing through $\langle s, o\rangle$ in the descending phase by using l and l' is obtained from the number of source–destination node pairs obtained from these sets.

$$\begin{aligned} C_b(\langle s, o\rangle, l, l') &= card(N_f^t(\langle s, o\rangle, l)) \times card(N_b^R(\langle s, o\rangle, l')) = \\ &= D_f^t(\langle s, o\rangle, l) \times D_b^R(\langle s, o\rangle, l') = (k^n - k^{s+1}) \times 1 = k^n - k^{s+1} \end{aligned}$$

□

Proposition 1.20 *Given the ports l and l' of the switch $\langle s, o\rangle$, where $0 \leq s < n-1$, $k \leq l < 2k$ and $0 \leq l' < k$, there are $k^n - k^{s+1}$ paths in the descending phase between l and l', if and only if $l' = l - k$.*

Proof: According to Proposition 1.19, at the most there are $k^n - k^{s+1}$ paths in the descending phase using the input port l and the output port l'. Similarly, according to Proposition 1.12 a path passes through $\langle s, o\rangle$ in the descending phase by using l and l' if $l' = l - k$ is satisfied, and therefore

$$C_b(\langle s, o\rangle, l, l') = \begin{cases} k^n - k^{s+1}, & \text{if and only if } l' = l - k \\ 0, & \text{otherwise} \end{cases}$$

□

Proposition 1.21 *The number of paths passing through the switch $\langle s, o\rangle$, $0 \leq s < n-1$, in the descending phase, is $k^{n+1} - k^{s+2}$.*

Proof: The total number of paths passing through the switch $\langle s, o\rangle$ in the descending phase is obtained from the multiplication of the number of paths that pass through $\langle s, o\rangle$ by using l and l' ($0 \leq l' < k$ and $k \leq l < 2k$) by the number of pairs (l,l'). Such pairs of ports satisfy $l' = l - k$, in other words, k.

$$C_b(\langle s, o\rangle, l, l') = C_b(\langle s, o\rangle, l, l') \times k = (k^n - k^{s+1}) \times k = k^{n+1} - k^{s+2}$$

that is

$$C_b(\langle s, o\rangle) = k^{n+1} - k^{s+2}$$

□

To sum up, Table 1 outlines the expressions obtained in the previous propositions.

Table 1: Number of paths passing through the switch $\langle s, o\rangle$ in the ascending, turnaround and descending phases under uniform traffic.

$$\begin{array}{rcl}
C_f(\langle s, o\rangle) & = & \begin{cases} k^{n+1} - k^{s+2}, & \text{if } s \in [0, n-2] \\ 0, & \text{if } s = n-1 \end{cases} \\
T(\langle s, o\rangle) & = & (k-1)k^{s+1} \\
C_b(\langle s, o\rangle) & = & \begin{cases} k^{n+1} - k^{s+2}, & \text{if } s \in [0, n-2] \\ 0, & \text{if } s = n-1 \end{cases} \\
C(\langle s, o\rangle) & = & \begin{cases} 2(k^{n+1} - k^{s+2}), & \text{if } s \in [0, n-2] \\ 0, & \text{if } s = n-1 \end{cases} \\
C_{f\ 0\leq l<k\ k\leq l'<2k}(\langle s, o\rangle, l, l') & = & \begin{cases} k^{n-1} - k^s, & \text{if } s \in [0, n-2] \\ 0, & \text{otherwise} \end{cases} \\
T_{0\leq l,l'<k\ l\neq l'}(\langle s, o\rangle, l, l') & = & k^s \\
C_{b\ k\leq l<2k\ 0\leq l'<k}(\langle s, o\rangle, l, l') & = & \begin{cases} k^n - k^{s+1}, & \text{if } s \in [0, n-2] \text{ and } l' = l - k \\ 0, & \text{otherwise} \end{cases}
\end{array}$$

5.2 Switch Classification

According to the expressions in Table 1, when the network topology is a N end nodes k–ary n–tree T–BMIN, the generated traffic is based on the uniform traffic pattern and the paths are determined by the routing algorithm defined in Section A.7.1, two types of switches are identified according to the connections required in the switches:

Type *va* Switches establishing forward, turnaround and backward connections belong to this type. The number of paths passing through a switch of this type are obtained by Propositions 1.14, 1.17 and 1.21. Similarly, the number of

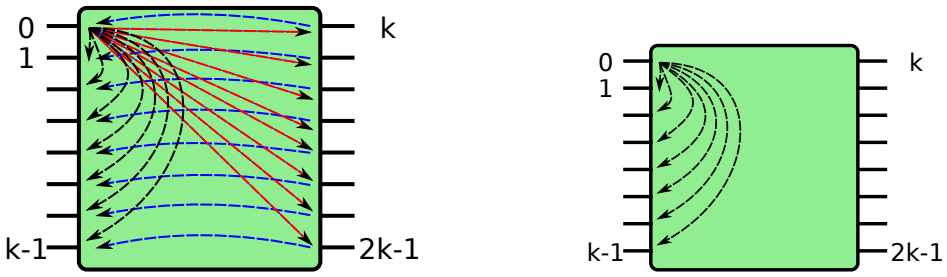
connections are determined by Propositions 1.13, 1.16, 1.19 and 1.20. All the switches in the BMIN are Type *va*, except those placed in the last stage.

Type *vb* Switches that only establish backward connections are Type *vb* switches according to Propositions 1.15, 1.17 and 1.18. The number of paths passing through one of these switches is determined by Proposition 1.17, and the number of required turnaround connections are obtained by the Proposition 1.16. The switches in the last stage are Type B.

When the paths are uniformly distributed between the ports of the switch, it is not helpful to provide a diagram with all the generated paths as it is done in the previous sections. The Figure 8 shows only the paths passing through the switch by using the input port zero in the ascending and turnaround phases. The remaining paths, whose input port is not equal to zero, have not been included for the sake of clarity. Moreover, the paths in the descending phase are independent from the other phases.

Figure 8(a) shows a Type *va* switch diagram, where it is shown the paths in the ascending, turnaround and descending phase in red, black and blue color, respectively.

Switches belonging to Type *vb* only have turnaround connections printed with black color. Figure 8(b) includes the paths from input port zero. The remaining paths from port 1 to $k-1$ are not drawn for clarity.



(a) Type *va*. (b) Type *vb*.

Figure 8: Connections required for each type of switch.

5.3 Switch Configuration

To apply the third step of the methodology and to find out the most adequate T-switch configuration for each type of switches, we decompose this step in turn in other two substeps:

a) For each type of network switch, and from the number of paths that use of the internal link, and the required connections by these paths in the switch, we will obtain separately the set of optimal configurations for each type of path (i.e. forward, turn-around and backward).

b) For each type of network switch, the optimal set of configurations, from the sets of optimal configurations previously obtained for each type of path, is obtained. In other words, the resulting set is that describes how to provide a $2k$-port switch $\langle s, o\rangle$ from two k-ports internal switches α and β.

This approach was chosen primarily for two reasons:

a) It is easier to address several simpler subproblems, from which the global solution can be derived (e.g., divide and conquer strategy) rather than to address the entire problem.

b) It can be done separately because the behavior of each case is independent of others. The cause because a connection uses or not the internal link only depends on T–switch configuration and is independent of the links to be used by other connections. Therefore, if a configuration minimizes separately the use of the internal link to both forward and backward connections, it also minimizes if we consider both types of connections at a time.

In short, the ultimate goal will be to calculate the set $\mathcal{S}^{\gamma}$ when under a concrete traffic pattern γ.

For the sake of notation and clarity, and different types of connections will be analyzed, we denote by using a superscript in $\mathcal{S}$ the link pattern applied, depending on the type of switch to be analyzed. We will use a subscript for indicating the path direction: b (backward), t (turn-around), f (forward), or a combination of them. When hidden, we would assume all the connections in the switch. For example, $\mathcal{S}_b^{\pi a}$ refers to the set of optimal T-switch configurations of type a under uniform traffic pattern(υ) and taken into account only the backward connections.

5.3.1 Type υa configuration of switch

According to such a methodology, we first obtain the optimal configurations to forward, turnaround and backward connections. In this case, the forward and turnaround connections are considered together, and then we consider separately the backward connections. Finally, we reason about optimal T–switch configurations taking into account all the connections.

5.3.1.1 Optimal switch configuration to forward and turnaround connections

Proposition 1.22 *Let $\mathcal{S}_{ft}^{\upsilon a}$ be the set of configurations that minimize the use of the internal link in a T–switch of type υa, considering the forward and turn-around connections, then*

$$\mathcal{S}_{ft}^{\upsilon a} = \{\mathcal{C} \in \mathcal{V} \mid card(\mathcal{B}) = card(\mathcal{F}) = k/2\}$$

and there exist $\frac{1}{2}k^{n+1}$ connections that use the internal link.

Proof: Given a port l, $0 \leq l < k$, the same number of forward connections are established with each port l', $k \leq l' < 2K$, and also the same number of turnaround connections with each port l'', $0 \leq l'' < k$ and $l \neq l''$. That is, every port l behave identically with independence of the switch.

Therefore, to obtain optimal configurations, it is not necessary to know what specific port is connected to a particular internal switch. It is enough to simply calculate how many upwards and downwards ports are connected to each internal switch. And this is the same as finding the optimal value of p.

However, this reasoning would not be useful if turnaround and forward connections are treated separately: the best configuration for turnaround connections ($p = k$), is the worst for forward connections, and vice versa. So, we treat both cases together to calculate the value of p that minimizes the number of paths passing through the internal link, considering the two types of connections.

To obtain p, we firstly calculate how many connections use the internal link that interconnects switches α and β. The resultant expression is dependent on p. Then we apply a mathematical procedure to know which value of p minimizes such as expression.

To know how many paths cross the internal link between α and β is used, C_i (Section 4.1), we must take into account the needed connections for any T–switch under uniform traffic, and the internal structure of the T–switch.

Under uniform traffic, it has been shown that paths in the forward and turnaround phases require at least one connection between any pair of ports l and l', $0 \leq l < k$ and $k \leq l' < 2K$, that is, a total of k^2 and $k^2 - k$ connections, respectively.

Under these conditions, that is, considering the k^2 possible forward connections in a switch, an expression for $CC_{If}(\langle s, o \rangle)$ was obtained (Section 4.1), which indicates the number of these connections that use the internal link connecting switches α and β. In the same way, and considering the $k^2 - k$ possible turnaround connections in the switch, an expression for $TC_I(\langle s, o \rangle)$ was obtained, which indicates the number of these connections also involving the use of the internal link connecting switches α and β.

Therefore, to obtain the contribution to $C_I(\langle s, o \rangle)$ of the forward and turnaround connections we properly combine the values indicated above for $CC_{If}(\langle s, o \rangle)$ and $TC_I(\langle s, o \rangle)$ with $C_f(\langle s, o \rangle)$ and $T(\langle s, o \rangle)$, respectively.

With all the above, the value of $C_I(\langle s, o\rangle)$ is determined as follows:

$$\begin{aligned} C_I(\langle s,o\rangle) &= \frac{C_f(\langle s,o\rangle)\times CC_{If}(\langle s,o\rangle)}{k^2} + \frac{T(\langle s,o\rangle)\times TC_I(\langle s,o\rangle)}{k^2-k} = \\ &= (k^{n-1}-k^s)\times(2p^2+k^2-2kp) + k^s\times 2(kp-p^2) = \\ &= 2p^2k^{n-1} - 2p^2k^s + k^{n+1} - k^{s+2} - 2pk^n + 2pk^{s+1} + \\ &\quad + 2pk^{s+1} - 2p^2k^s = \\ &= 2p^2k^{n-1} - 4p^2k^s + k^{n+1} - k^{s+2} - 2pk^n + 4pk^{s+1} \end{aligned}$$

To find out the p value that minimizes $C_I(\langle s, o\rangle)$, it is required to determine the derivative of $C_I(\langle s, o\rangle)$ with respect to p.

$$C'_I(\langle s,o\rangle) = \frac{\partial}{\partial p} C_I(\langle s,o\rangle) = 4pk^{n-1} - 8pk^s - 2k^n + 4k^{s+1}$$

The derivative is equal to zero, $C'_I(\langle s, o\rangle) = 0$, to calculate the maximum or minimum points.

$$\begin{aligned} C'_I(\langle s,o\rangle) &= 0 \\ 4pk^{n-1} - 8pk^s - 2k^n + 4k^{s+1} &= 0 \\ 4pk^{n-1} - 8pk^s &= 2k^n - 4k^{s+1} \\ p(4k^{n-1} - 8k^s) &= 2k^n - 4k^{s+1} \\ p &= \frac{2k^n - 4k^{s+1}}{4k^{n-1} - 8k^s} \end{aligned}$$

As $0 \leq s < n-1$, the expression can be simplified by removing the common factor k^s

$$p = \frac{k^s(2k^{n-s} - 4k)}{k^s(4k^{n-s-1} - 8)} = \frac{2k^{n-s} - 4k}{4k^{n-s-1} - 8} = \frac{k(2k^{n-s-1} - 4)}{2(2k^{n-s-1} - 4)} = \frac{k}{2}$$

To know if $C_I(\langle s, o\rangle)$ reaches a maximum or minimum point in $p = k/2$, the second derivative of $C_I(\langle s, o\rangle)$ with respect to p is calculated, and it is checked its value for $p = k/2$. If the sign of the second derivative is negative or positive, the function will have a maximum or minimum in $p = k/2$, respectively. But the second derivative may be zero at an inflection point (e.g. a point where the second derivative of a function changes sign is called an inflection point).

$$C''_I(\langle s,o\rangle) = \frac{\partial}{\partial p} C'_I(\langle s,o\rangle) = 4k^{n-1} - 8k^s = 4k^s(k^{(n-1)-s} - 2)$$

Since $s < n-1$, $(n-1) - s \geq 1$. So, the sign of $C''_I(\langle s, o\rangle)$ is positive for $k > 2$, which is true under the initial hypotheses (Section 2.3). Therefore, $p = k/2$ is a

minimum for the function $C_I(\langle s, o\rangle)$. Substituting the p value in $C_I(\langle s, o\rangle)$, we obtain that:

$$\begin{aligned} C_I(\langle s, o\rangle) &= 2\left(\frac{k}{2}\right)^2 k^{n-1} - 4\left(\frac{k}{2}\right)^2 k^s + k^{n+1} - k^{s+2} - 2\frac{k}{2}k^n + 4\frac{k}{2}k^{s+1} = \\ &= \frac{1}{2}k^{n+1} - k^{s+2} + k^{n+1} - k^{s+2} - k^{n+1} + 2k^{s+2} = \\ &= \frac{1}{2}k^{n+1} \end{aligned}$$

That is to say, a configuration is optimum if $p = q = k/2$. However, the number of paths crossing the internal link of the T–switch is $\frac{1}{2}k^{n+1}$. □

Figure 9 is a general scheme of connections for the optimal configuration of a 8×8 T–switch.

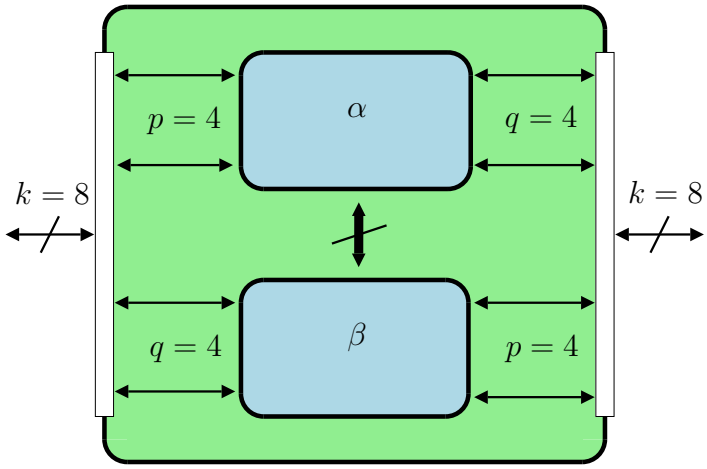


Figure 9: Optimal configuration of a 8×8 T–switch of type va considering forward and turnaround connections.

Example 1.5 shows an example of optimal configuration for T–switches, when both forward and turnaround connections are considered.

Example 1.5 *Given the configuration $\mathcal{T}$ of a 8×8 T–switch such that*

$$\mathcal{T} = \left\{\mathcal{C}^\alpha, \mathcal{C}^\beta \in \mathcal{V} \mid \mathcal{C}^\alpha = \{0,1,2,3,8,9,12,13\},\ \mathcal{C}^\beta = (\mathcal{C}^\alpha)^C = \{4,5,6,7,10,11,14,15\}\right\}$$

$\mathcal{T}$ is an optimal configuration to forward and turnaround connections of a T–switch of type va, because it is verified that:

$$\mathcal{C}^\alpha,\ \mathcal{C}^\beta \in \mathcal{S}_{ft}^{va}$$

In particular, the connections of $\mathcal{T}$ are shown in the Figure 10.

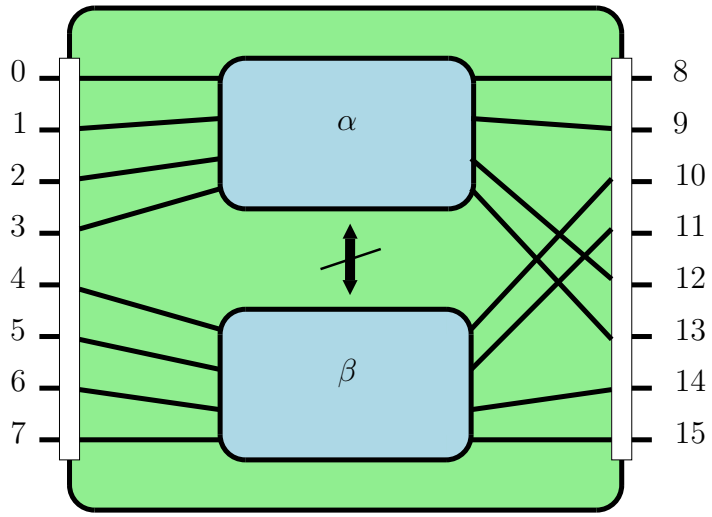


Figure 10: Example of optimal configuration for a 8×8 T–switch of type va considering the forward and turnaround connections.

5.3.1.2 Optimal configuration to backward connections By Proposition 1.12, there exist backward connections between a port l, $k \leq l < 2k$, and another port l', $0 \leq l' < k$, if and only if $l' = l - k$. Following a similar procedure to previous one in Section 5.3.1.1, the binary relation $\mathcal{R}_b$ is defined, and its corresponding Propositions, from which the set of optimal configurations for backward connections in a T–switch is derived. These configurations have also the property of being optimum for whatever traffic pattern generated in the network.

Definition 1.22 *The binary relation $\mathcal{R}_b$ on a set $\mathcal{U}$ is defined as follows*

$$\mathcal{R}_b = \{(l, l') \in \mathcal{U}^2 \mid |l - l'| = k \}$$

Note that expression $|l - l'| = k$ comes from $l' = l - k$ because

$$l' = l - k \Rightarrow \begin{cases} l' - l = -k \\ l - l' = k \end{cases} \Rightarrow \begin{cases} |l' - l| = |-k| \\ |l - l'| = |k| \end{cases} \Rightarrow |l' - l| = |l - l'| = k$$

Proposition 1.23 *$\mathcal{R}_b$ is a symmetric relation.*

Proof: If $\mathcal{R}_b$ is symmetric, then it will verify that $\forall l, l' \in \mathcal{U}, (l, l') \in \mathcal{R}_b \Rightarrow (l', l) \in \mathcal{R}_b$. The demonstration is trivial because

$$|l - l'| = |-(l - l')| = |l' - l| = k$$

consequently $\mathcal{R}_b$ is symmetric. □

Proposition 1.24 *Let $l, l', l'' $ be three ports such that $l, l', l'' \in \mathcal{U}$ and $(l, l') \in \mathcal{R}_b$. It is verified that $(l, l'') \in \mathcal{R}_b$ if and only if $l' = l''$.*

Proof: If $|l - l'| = k$ then

$$|l - l'| = k \Rightarrow \begin{cases} l - l' = k & \Rightarrow \quad l' = l - k \\ & \text{or} \\ l - l' = -k & \Rightarrow \quad l' = l + k \end{cases}$$

Although there are two possible values for l', only one is a valid port. It is clear that if $0 \leq l < k$, then $l' = l - k < 0$ is not a valid port because it is out of bounds of valid ports; otherwise, if $k \leq l < 2k$, then $l' = l + k \geq 2k$ will be an invalid port. In any case, there will be one port $l' \in \mathcal{U}$ such that $(l, l') \in \mathcal{R}_b$.

On the other hand, if $l' = l''$, then it will be trivial to demonstrate that $(l, l'') \in \mathcal{R}_b$. □

The Propositions 1.25 and 1.26 that are enunciated and demonstrated bellow, are not only applicable under uniform traffic, but under any traffic pattern, since they are only derived from the network topology and routing algorithm.

Proposition 1.25 *Let $\mathcal{S}_b$ be the set of configurations that minimize the use of the internal link in a T–switch considering the backward connections, then*

$$\mathcal{S}_b = \{\mathcal{C} \in \mathcal{V} \mid \forall l \in \mathcal{C},\ \exists l' \in \mathcal{C} \mid (l, l') \in \mathcal{R}_b\}$$

and there are no backward connections that go across the internal link.

Proof: Let $\mathcal{C}$ be a configuration such that $\mathcal{C} \in \mathcal{S}_b$. According to the definition of $\mathcal{S}_b$, for a port $l \in \mathcal{C}$ there exists another port $l' \in \mathcal{C}$ such that $(l, l') \in \mathcal{R}_b$. This in turn implies that there exists a port $l'' \in \mathcal{C}$ such that $(l', l'') \in \mathcal{R}_b$; and another port $l''' \in \mathcal{C}$ such that $(l'', l''') \in \mathcal{R}_b$; and so on.

Nevertheless, according to Propositions 1.23 and 1.24, the inclusion of port l in $\mathcal{C}$ implies the inclusion of a unique port l' satisfying $(l, l') \in \mathcal{R}_b$.

Moreover, since $card(\mathcal{C}) = k$ (Proposition 1.2), k is even (initial hypothesis); and all the k ports are grouped in pairs (l', l'') which verify the binary relation $\mathcal{R}_b$, there would be always configurations belonging to $\mathcal{S}_b$. Hence, $\mathcal{S}_b \neq \emptyset$.

On the other hand, there are only connections between two ports l and l', if $(l, l') \in \mathcal{R}_b$. If $\forall l \in \mathcal{C}$, there exists a port l' such that $(l, l') \in \mathcal{R}_b$, then all the connections are established between port belonging to the same internal switch. Consequently, there exist no connections that use the internal link; and the members of $\mathcal{S}_b$ are optimal configurations. □

Proposition 1.26 *If there exist backward connections for every pair of ports l and l', such that $(l, l') \in \mathcal{R}_b$, then the members belonging to $\mathcal{S}_b$ are the unique optimal configurations. Hence,*

$$\forall (l, l') \in \mathcal{R}_b,\ C_b(\langle s, o\rangle, l, l') > 0 \Rightarrow \forall \mathcal{C} \in \mathcal{V},\ \textit{if } \mathcal{C} \notin \mathcal{S}_b,\ \textit{then } \mathcal{C} \textit{ is not optimum}$$

Proof: Assuming the existence of backward connections for all the pairs of ports l and l' that verify $(l, l') \in \mathcal{R}_b$, let us suppose that there exists a configuration $\mathcal{C}'$ that minimizes the use of the internal link, and $\mathcal{C}' \notin \mathcal{S}_b$. If $\mathcal{C}' \notin \mathcal{S}_b$, then there will exist a port l that is not related to l'.

$$\mathcal{C}' \notin \mathcal{S}_b \ \Rightarrow\ \exists l \in \mathcal{C}' \mid \forall l' \in \mathcal{C}',\ (l, l') \notin \mathcal{R}_b$$

If the configuration $\mathcal{C}'$ is used to build a T–switch (it should be reminded $\mathcal{C}'$ determines the ports connected to each internal switch) the port l must accomplish the backward connections to a port that is not allocated inside the same internal switch, since all the ports establish backward connections. Consequently, those connections have to use the internal link, and the configuration $\mathcal{C}'$ does not minimize the use of the internal link, since there are configurations better than $\mathcal{C}'$ (because they do not go across the internal link), which belong to $\mathcal{S}_b$.

Therefore, the members of $\mathcal{S}_b$ are configurations that minimize the number of connections that use the internal link. □

Example 1.6 shows two optimal configurations for a T–switch of type πa considering the backward connections.

Example 1.6 *Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be two configurations for a 8×8 T–switch*

$$\mathcal{T}_1 = \left\{\mathcal{C}_1^{\alpha}, \mathcal{C}_1^{\beta} \in \mathcal{V} \mid \mathcal{C}_1^{\alpha} = \{0,1,2,3,8,9,10,11\},\ \mathcal{C}_1^{\beta} = (\mathcal{C}_1^{\alpha})^C = \{4,5,6,7,12,13,14,15\}\right\}$$

$$\mathcal{T}_2 = \left\{\mathcal{C}_2^{\alpha}, \mathcal{C}_2^{\beta} \in \mathcal{V} \mid \mathcal{C}_2^{\alpha} = \{0,2,3,4,8,10,11,12\},\ \mathcal{C}_2^{\beta} = (\mathcal{C}_2^{\alpha})^C = \{1,5,6,7,9,13,14,15\}\right\}$$

Both $\mathcal{T}_1$ and $\mathcal{T}_2$ are optimal configurations for the backward connections in a T–switch because they verify

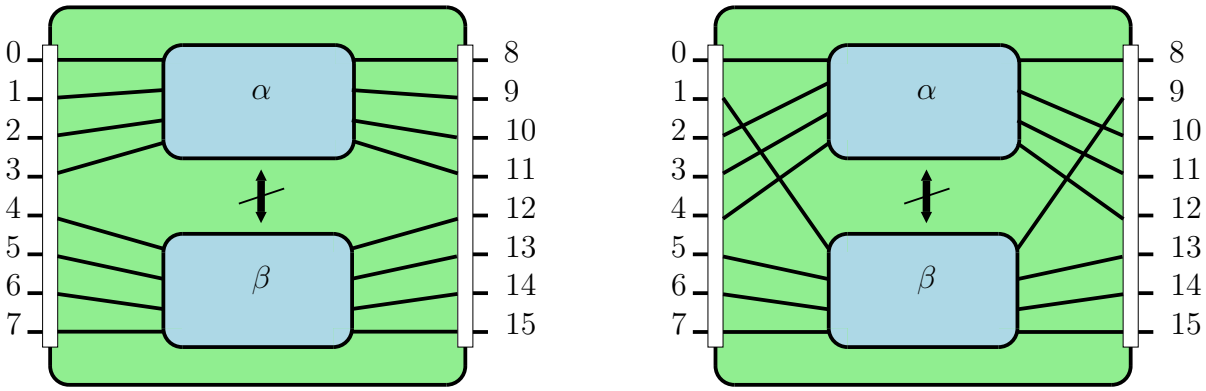
$$\mathcal{C}_1^{\alpha},\ \mathcal{C}_1^{\beta},\ \mathcal{C}_2^{\alpha},\ \mathcal{C}_2^{\beta} \in \mathcal{S}_b$$

The Figures 11a and 11b illustrate the connections for the configurations $\mathcal{T}_1$ and $\mathcal{T}_2$, respectively.

Proposition 1.27 *Let $\mathcal{S}_b^{\pi a}$ be the set of configurations that minimize the use of the internal link in a T–switch of type πa, considering the backward connections, then*

$$\mathcal{S}_b^{\pi a} = \mathcal{S}_b = \{\mathcal{C} \in \mathcal{V} \mid \forall l \in \mathcal{C},\ \exists l' \in \mathcal{C} \mid (l, l') \in \mathcal{R}_b\}$$

and there are no backward connections that go across the internal link.



(a) $\mathcal{T}_1$ (b) $\mathcal{T}_2$

Figure 11: Optimal switch configurations for a 8×8 T–switch considering backward connections.

Proof: From Proposition 1.25, members of $\mathcal{S}_b$ are optimal configurations considering backward connections. Moreover, in a T–switch of type πa, where there exist backward connections between each pair of ports l, l' such that $(l, l') \in \mathcal{R}_b$, the configurations belonging to $\mathcal{S}_b$ are the unique optimal configurations by the Proposition 1.26. □

Proposition 1.28 *Let $\mathcal{S}_b^{va}$ be the set of configurations that minimize the use of the internal link in a T–switch of type va, considering the backward connections, then*

$$\mathcal{S}_b^{va} = \mathcal{S}_b = \{\mathcal{C} \in \mathcal{V} \mid \forall l \in \mathcal{C},\ \exists l' \in \mathcal{C} \mid (l, l') \in \mathcal{R}_b\}$$

and there no backward connections that go across the internal link.

Proof: By Proposition 1.25, the configurations in $\mathcal{S}_b$ are optimal to backward connections. Moreover, in a T–switch of type va there exist backward connections between a pair of ports l, l' if $(l, l') \in \mathcal{R}_b$. According to Proposition 1.26, we know the configurations belonging to $\mathcal{S}_b$ are the only optimal configurations. □

The Example 1.7 shows two optimal configurations for a T–switch of type va only considering backward connections.

Example 1.7 *Given two configurations, $\mathcal{T}_1$ and $\mathcal{T}_2$, for a 8×8 T–switch such that*

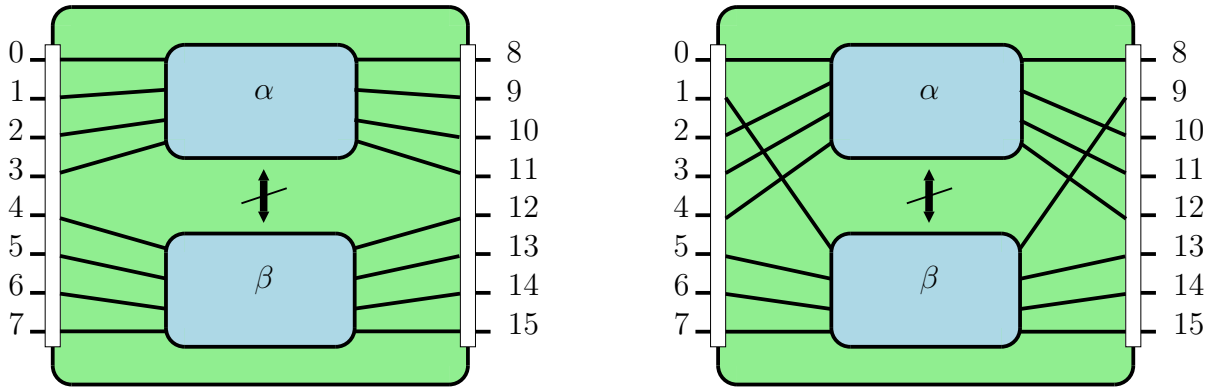
$$\mathcal{T}_1 = \left\{\mathcal{C}_1^{\alpha}, \mathcal{C}_1^{\beta} \in \mathcal{V} \mid \mathcal{C}_1^{\alpha} = \{0,1,2,3,8,9,10,11\},\ \mathcal{C}_1^{\beta} = (\mathcal{C}_1^{\alpha})^C = \{4,5,6,7,12,13,14,15\}\right\}$$

$$\mathcal{T}_2 = \left\{\mathcal{C}_2^{\alpha}, \mathcal{C}_2^{\beta} \in \mathcal{V} \mid \mathcal{C}_2^{\alpha} = \{0,2,3,4,8,10,11,12\},\ \mathcal{C}_2^{\beta} = (\mathcal{C}_2^{\alpha})^C = \{1,5,6,7,9,13,14,15\}\right\}$$

Both $\mathcal{T}_1$ and $\mathcal{T}_2$ are optimal configurations to backward connections in a T–switch of type va, because it is verified that:

$$\mathcal{C}_1^{\alpha},\ \mathcal{C}_1^{\beta},\ \mathcal{C}_2^{\alpha},\ \mathcal{C}_2^{\beta} \in \mathcal{S}_b^{va}$$

In Figure 12, the connections of the configurations $\mathcal{T}_1$ and $\mathcal{T}_2$ are shown.



(a) $\mathcal{T}_1$ (b) $\mathcal{T}_2$

Figure 12: Possible optimal configurations for a 8×8 T–switch of type va considering backward connections.

5.3.1.3 Optimal configuration to all the connections

Proposition 1.29 *Let $\mathcal{S}^{va}$ be the set of configurations that minimize the use of the internal link in a T–switch of type va, then*

$$\mathcal{S}^{va} = \{\mathcal{C} \in \mathcal{V} \mid \forall l \in \mathcal{C},\ \exists l' \in \mathcal{C} \mid (l, l') \in \mathcal{R}_b\}$$

and there are $\frac{1}{2}k^{n+1}$ connections going across the internal link.

Proof: To obtain the configurations that minimize the use of the internal link considering all the connections, the intersection between the sets that define the optimal configurations considering forward and turnaround, and backward connections, respectively, is applied.

$$\begin{aligned}\mathcal{S}^{va} = \mathcal{S}^{va}_{ft} \cap \mathcal{S}^{va}_{b} = \{\mathcal{C} \in \mathcal{V} \mid card(\mathcal{B}) = card(\mathcal{F}) = k/2\} \cap \\ \cap \{\mathcal{C} \in \mathcal{V} \mid \forall l \in \mathcal{C},\ \exists l' \in \mathcal{C} \mid (l, l') \in \mathcal{R}_b\} = \\ \{\mathcal{C} \in \mathcal{V} \mid card(\mathcal{B}) = card(\mathcal{F}) = k/2, \forall l \in \mathcal{C},\ \exists l' \in \mathcal{C} \mid (l, l') \in \mathcal{R}_b\}\end{aligned}$$

The condition imposed by $\mathcal{S}^{va}_b$ meets in turn the condition imposed by $\mathcal{S}^{va}_{ft}$. Let $l, l' \in \mathcal{C}$ be two ports such that $(l, l') \in \mathcal{R}_b$ (i.e. $|l - l'| = k$). Then

$$|l - l'| = k \Rightarrow \begin{cases} l - l' = k & \Rightarrow \quad l' = l - k \\ & \text{or} \\ l - l' = -k & \Rightarrow \quad l' = l + k \end{cases}$$

If $l \in \mathcal{B}$, then $0 \leq l < k$ and $l' = l + k$, because if l' were equal to $l - k$, then $l' < 0$ and $l' \notin \mathcal{U}$ would meet. Consequently, $k \leq l' < 2k$ and $l' \in \mathcal{F}$. However, if $l \in \mathcal{F}$, then $k \leq l < 2k$ and $l' = l - k$, because if l' were equal to $l + k$, then $l' \geq 2k$ and $l' \notin \mathcal{U}$ would meet. Therefore, $0 \leq l' < k$ and $l' \in \mathcal{B}$. That is to say, for each port belonging to $\mathcal{B}$, the set $\mathcal{C}$ contains another port belonging to $\mathcal{F}$.

In other words, $\mathcal{B}$ and $\mathcal{F}$ have the same number of ports. On the other hand, from the Proposition 1.2, $card(\mathcal{C}) = k$, and then, it is verified that $card(\mathcal{B}) = card(\mathcal{F}) = card(\mathcal{C})/2 = k/2$. Thus, the first condition can be eliminated because it is redundant, and we have:

$$\mathcal{S}^{va} = \{\mathcal{C} \in \mathcal{V} \mid \forall l \in \mathcal{C},\ \exists l' \in \mathcal{C} \mid l' = (l, l') \in \mathcal{R}_b\}$$

With these configurations, a total of $\frac{1}{2}k^{n+1}$ forward and turnaround connections require the use of the internal link, and there are no backward connections using the internal link. Consequently, if all the connections are established, the internal link will be only used by forward and turnaround connections, being $\frac{1}{2}k^{n+1}$ the total number of them.

In the following, we prove by reductio ad absurdum that configurations in $\mathcal{S}^{va}$ are the unique optimal configurations. Let us assume an arbitrary configuration, $\mathcal{C}'$, that minimizes the use of the internal link and $\mathcal{C}' \notin \mathcal{S}^{va}$. This implies there exists a port l that is not related to another port l'.

$$\mathcal{C}' \notin \mathcal{S}^{va} \ \Rightarrow\ \exists l \in \mathcal{C}' \mid \forall l' \in \mathcal{C}'(l, l') \notin \mathcal{R}_b$$

In such a case, the $k^n - k^{s+1}$ backward connections using the port l have to pass through the internal link. On the assumption that C' still verifies $card(\mathcal{B}) = card(\mathcal{F}) = k/2$, there exist $\frac{1}{2}k^{n+1}$ forward and turnaround connections using the internal link, and the total number of connections that use the internal link is $\frac{1}{2}k^{n+1} + k^n - k^{s+1}$.

All the switches in the network are T–switches of type va, except the last–stage switches. That is,

$$0 \leq s < n - 1$$
$$1 \leq s + 1 < n$$

then

$$k^n > k^{s+1}$$
$$k^n - k^{s+1} > 0$$

and therefore

$$\frac{1}{2}k^{n+1} + k^n - k^{s+1} > \frac{1}{2}k^{n+1}$$

so $\mathcal{C}'$ does not minimizes the use of the internal link, because the connections crossing the internal link and belonging to $\mathcal{S}^{va}$ are $\frac{1}{2}k^{n+1}$ in total. This contradicts the assumption that $\mathcal{C}'$ minimizes the use of the internal link.

Therefore, the connections in $\mathcal{S}^{va}$ minimize the number of connections that use the internal link, and also, they are the unique ones. □

Since the set of optimal configurations obtained is the same as considering only backward connections, the configurations shown in the Example 1.7 are equally valid in this case.

5.3.2 Type vb configuration of switch

According to BMIN definition, only backward connections are established in the last–stage switches. Therefore, half the ports are used, that is, k ports. As the total number of ports in the internal switches coincides with k, one obvious and optimal configuration in the last–stage switches is that connecting all the k ports to only one internal switch. Two configurations are possible: $(p = k, q = 0)$ and $(p = 0, q = k)$ have similar behavior, that is, the internal link between the switches α and β is never used. In a more formal way:

Proposition 1.30 *Let $\mathcal{S}^{vb}$ be the set of configurations that minimize the use of the internal link in a T–switch of type vb, then*

$$\mathcal{S}^{vb} = \{\mathcal{C} \in \mathcal{V} \mid \mathcal{B} = \emptyset \text{ or } \mathcal{F} = \emptyset\}$$

and there are no connections using the internal link.

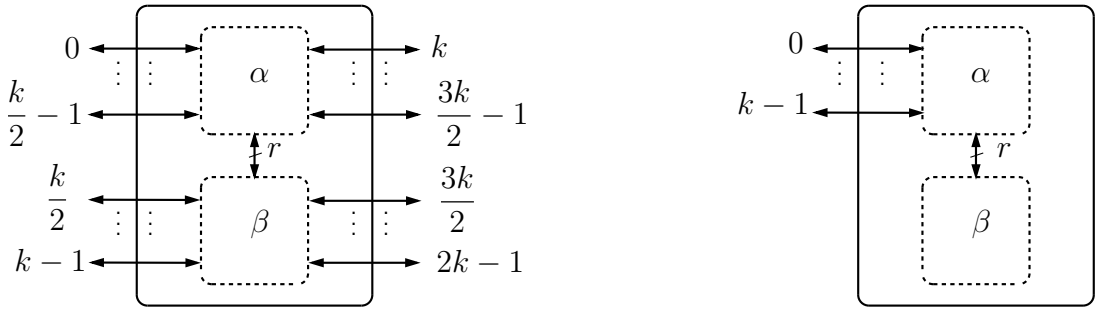
Proof: If $\mathcal{B} = \emptyset$ then $card(\mathcal{B}) = 0$, and by Definition 1.8 and according to the Proposition 1.2 ($card(\mathcal{C}) = k$) is verified that $card(\mathcal{F}) = k$. This implies that all the forward ports are connected to $\mathcal{C}$, and then, all the backward ports are connected to $\mathcal{C}^C$. Since to find out a configuration $\mathcal{T}$ of a T–switch is necessary to take into account $\mathcal{C}$ and $\mathcal{C}^C$, all the backward links of the T–switch are obtained from all the ports of a single internal switch.

On the other hand, if $\mathcal{F} = \emptyset$ then $card(\mathcal{F}) = 0$ and by the Definition 1.8 and according to the Proposition 1.2 it is verified that $card(\mathcal{B}) = k$. Similar to the previous case, all the backward ports of T–switch are obtained from a single internal switch.

It is clear if all the backward links of the T–switch are obtained with a single internal switch, no turnaround connections will use the internal link that interconnects the internal switches of the T–switch. □

5.3.3 Configuration of switch

According to Section 5.3, the optimal T-switch configuration depends on the specific type of T-switch. Based on the previous sections 5.3.1 and 5.3.2, all the T-switches $\langle s, o\rangle$, $0 \leq s < n-1$, in the BMIN can be setup with an Type va configuration because they are past through by paths in the ascending, turnaround, and descending phase. At the last-stage switches, the optimal T-switch configuration is the optimal Type vb configuration because they are uniquely past through by paths in the turnaround phase. Both configuration are shown in Figure 13.



(a) T-switches that are placed in the stage s where $0 \leq s < n-1$.

(b) T-switches that are placed in the last stage $s = n-1$.

Figure 13: Optimal switch-level connection pattern ($OSCP$).

6 High-radix Switches

In this section we review existing proposals of high-radix switches in the literature, which are mainly focused on solving the scaling problems from traditional switch designs.

The switch YARC is the switch high-radix used by the Cray BlackWidow [SAKD06]. BlackWidow uses a folded-Clos topology. YARC is a hierarchical switch with the internal organization defined in [KDTG05], which states that increasing the level of the switch is the most efficient strategy to increase the bandwidth of the switch. The authors claim that for a specific number of signals in the chip is preferable to form more links (giving fewer signals per link) that implement fewer links at the expense of allocating more signals to each link.

The paper makes clear that the traditional designs of switches with fewer ports can not be adapted to the new switches with many ports, which refer as high-radix, because the traditional centralized organization does not scale properly. The report proposes a new hierarchical switch architecture that improves the performance of traditional switches with few number ports. The new architecture distributes and simplifies the control logic and reduces the communication lines within the chip. The result is a viable switch, but with a very low yield, due to the problem of head of line blocking. Adding buffers in the crossing points at the crossbar can eliminate the HOL blocking decoupling the input and output of the switch.

The *folded-Clos* topology doubles in cost to the butterfly multistage topology with the same capacity and higher latency than a butterfly bidirectional multistage network. This is because in the butterfly multistage packets are sent through intermediate stages of the network before being routed to their final destination. The topology *flattened butterfly* [KDA07] is an alternative to the folded-Clos topology. Improvements in the signal technology have allowed longer cable lengths. Based on the conventional butterfly multistage topology, the switches of the intermediate stages are replaced by high-radix switches and they are connected with new longer cables. As a result, it reduces the number of jumps for intermediate switches, which decreases the latency. In addition, fewer switches means less cost.

Overlooking the advantages of high-radix switches, *Dragonfly* [KDSA08] topology proposes to increase the effective radix of the switch using a set of switches interconnected by a subnet. The set operates as a virtual switch within a hierarchical network. The hierarchy has three levels: the lower level which nodes connect to the virtual switches at the intermediate level, there is a local subnet switches interconnecting the intermediate level, and finally there is a global subnet to interconnect virtual switches.

The three topologies considered that high-radix switches are homogeneous elements. All exploit the number of available ports, but none discusses the possible variation of network performance based on which ports are used to connect the different components of the network.

The *Partitioned Crossbar Input Queued (PCIQ)* switch [MFD+06] is a more recent proposal for the internal organization of high-radix switches. It is based on replacing the central crossbar by several *internal* crossbars improving the readability of the buffers without increasing the hardware cost. Each crossbar has a round-robin arbiter, which has a linear cost and logarithmic response time, as the radix of the switch increases.

Moreover, the switch implements RECN [GFD+06] as congestion control mechanism. Thus, the switch completely eliminates the HOL blocking at both switch and network levels, thus the maximum productivity is not consistent with traffic. This architecture has a lower cost than the hierarchical architecture proposed in [KDTG05].

Additionally, for instance, a high-radix switch based network is required in order to exploit the computing power of a system made of Merrimac® stream processors [DHE+03]. Also new communication technologies like Proximity Communication (PxC) from Sun™ Microsystems are tied to high-radix architectural designs [EGF+08].

Regarding the alternative for building high-radix switches using low-radix switches, the Oracles's Sun™ Blade 6048 Infiniband QDR Switched Network Express Module (NEM) [Net10] has already implemented this strategy, offering the ability to connect up to 12 dual-node blades in a single shelf. Each NEM provides 12 connections from each of the 36-port Mellanox's InfiniScale® IV switches. A total of 24 connections are used to communicate with the two compute nodes on each of the dual-node server blades, with 9 ports used to connect the two switches together. The 30 remaining ports (15 per switch chip) are used as links to either other NEMs, or to external switches.

To the best of our knowledge, there are currently no formal studies published on determination of switch-level connection pattern.

This way to get high-radix switches requires a thorough study to find out the best switch-level connection pattern that we have done in this report and we are presenting from Section 2.2.

References

[aa89] H.J. Siegel at al. Using the multistage cube network topology in parallel supercomputers. In *Proceedings of the IEEE, vol. 77, pp. 1932–1953*, December 1989.

[DeH90] André DeHon. Technical report: Fat–tree routing for transit. Technical report, 1990.

[DHE+03] W. J. Dally, P. Hanrahan, M. Erez, T. J. Knight, F. Labonte, J-H A., N. Jayasena, U. J. Kapasi, A. Das, J. Gummaraju, and I. Buck. Merrimac: Supercomputing with streams. In *SC'03*, Phoenix, Arizona, November 2003.

[DYN03] José Duato, Sudhakar Yalamanchili, and Lionel Ni. *Interconnection networks. An engineering approach.* Morgan Kaufmann Publishers Inc., 2003.

[ea91] E. Bakker et al. Linear interval routing algorithms review 2., 1991.

[EGF+08] Hans Eberle, Pedro J. Garcia, José Flich, José Duato, Robert Drost, Nils Gura, David Hopkins, and Wladek Olesinski. High-radix crossbar switches enabled by proximity communication. In *SC '08: Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, pages 1–12, Piscataway, NJ, USA, 2008. IEEE Press.

[fSI10] International Technology Roadmap for Semiconductors (ITRS). International technology roadmap for semiconductors: 2010 update, 2010. `www.itrs.net/Links/2010ITRS/Home2010.htm`.

[GAG+03] M. Gusat, F. Abel, F. Gramsamer, R. Luijten, C. Minkenberg, and M. Verhappen. Stability degree of switches with finite buffers and non-negligible round-trip time[small star, filled]. *Microprocessors and Microsystems*, 27(5-6):243 – 252, 2003. International Conference on Computer, Communication and Networking 2002.

[GFD+06] P. J. García, J. Flich, J. Duato, I. Johnson, F. J. Quiles, and F. Naven. Efficient, scalable congestion management for interconnection networks. *IEEE Micro*, 26, 2006(5):52–66, September 2006.

[GGG+07] Crispín Gómez, Francisco Gilabert, María E. Gómez, Pedro Lopez, and José Duato. Deterministic versus adaptive routing in fat-trees. Los Alamitos, CA, USA, 2007. IEEE Computer Society.

[GL73] L. Rodney Goke and G. J. Lipovski. Banyan networks for partitioning multiprocessor systems. In *ISCA '73: Proceedings of the 1st annual symposium on Computer architecture*, pages 21–28, New York, NY, USA, 1973. ACM.

[GLD05] M. E. Gomez, P. Lopez, and J. Duato. A memory-effective routing strategy for regular interconnection networks. In *IPDPS '05: Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05) - Papers*, page 41.2, Washington, DC, USA, 2005. IEEE Computer Society.

[KDA07] John Kim, William J. Dally, and Dennis Abts. Flattened butterfly: a cost-efficient topology for high-radix networks. In *ISCA '07: Proceedings of the 34th annual international symposium on Computer architecture*, pages 126–137, New York, NY, USA, 2007. ACM.

[KDSA08] John Kim, Wiliam J. Dally, Steve Scott, and Dennis Abts. Technology-driven, highly-scalable dragonfly topology. In *ISCA '08: Proceedings of the 35th Annual International Symposium on Computer Architecture*, pages 77–88, Washington, DC, USA, 2008. IEEE Computer Society.

[KDTG05] John Kim, William J. Dally, Brian Towles, and Amit K. Gupta. Microarchitecture of a high-radix router. *SIGARCH Comput. Archit. News*, 33(2):420–431, 2005.

[KS83] C. Kruskal and M. Snir. The performance of multistage interconnection networks for multiprocessors. *IEEE Transactions on Computers*, C-32(12):1091–1098, December 1983.

[Lei85] C. E. Leiserson. Fat-trees: universal networks for hardware-efficient supercomputing. *IEEE Transactions on Computers*, 34(10):892–901, 1985.

[Lei92] F.T. Leighton. *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes*. Morgan Kaufmann Publishers, 1992.

[LM88] Charles Leiserson and Bruce M. Maggs. Communication-efficient parallel algorithms for distributed random-access machines. *Algorithmica*, 3:53–77, 1988.

[lWF80] Chuan lin Wu and Tse-Yun Feng. On a class of multistage interconnection networks. *IEEE Trans. Computers*, 29(8):694–702, 1980.

[MAM+05] Cyriel Minkenberg, Francois Abel, Peter Muller, Raj Krishnamurthy, Mitchell Gusat, and B. Roe Hemenway. Control path implementation for a low-latency optical hpc switch. In *Proceedings of the 13th Symposium on High Performance Interconnects*, HOTI '05, pages 29–35, Washington, DC, USA, 2005. IEEE Computer Society.

[MFD+06] G. Mora, J. Flich, J. Duato, P. López, E. Baydal, and O. Lysne. Towards an efficient switch architecture for high-radix switches. In *ANCS '06: Proceedings of the 2006 ACM/IEEE symposium on Architecture for networking and communications systems*, pages 11–20, New York, NY, USA, 2006. ACM.

[MG07] Cyriel Minkenberg and Mitchell Gusat. Speculative flow control for high-radix datacenter interconnect routers. *Parallel and Distributed Processing Symposium, International*, 0:70, 2007.

[Net10] Sun Networking. Sun datacenter infiniband switch 36, sun datacenter infiniband switch 72, sun datacenter infiniband switch 648: Architecture and deployment, April 2010.

[NGM97] L.M. Ni, Y. Gui, and S. Moore. Performance evaluation of switch-based wormhole networks. 8(5):462–474, May 1997.

[Pat81] J.H. Patel. Performance of processor-memory interconnections for multiprocessors. *IEEE Transactions on Computers*, C-30(10):771–780, October 1981.

[SAKD06] Steve Scott, Dennis Abts, John Kim, and William J. Dally. The blackwidow high-radix clos network. *SIGARCH Comput. Archit. News*, 34(2):16–28, 2006.

[SJS08] Frank Olaf Sem-Jacobsen and Tor Skeie. Maintaining quality of service with dynamic fault tolerance in fat-trees. In *HiPC*, pages 451–464, 2008.

[SS10] Top500 Supercomputer Site. `www.top500.org`, November 2010.

[WPM03] Hangsheng Wang, Li-Shiuan Peh, and Sharad Malik. Power-driven design of router microarchitectures in on-chip networks. In *Proceedings of the 36th annual IEEE/ACM International Symposium on Microarchitecture*, MICRO 36, pages 105–, Washington, DC, USA, 2003. IEEE Computer Society.

A Multistage Interconnection Networks

This appendix introduces multistage interconnection networks and presents some of their basic aspects, which will be used in the rest of this report. Fat–trees networks receive an special attention because they have been chosen for carrying out this study, due to they are one of the most used interconnection network in the supercomputers market.

A.1 Multistage interconnection networks

Multistage Interconnection Networks (MINS) connect input devices to output devices through a number of switch stages, where each switch is a crossbar network. The number of stages and the connections patterns between two adjacent stages determine the routing capability of the networks (Figure 14). Other characteristics considered in MINs are the number of switches and the switch radix, the number of stages, message average latency, path diversity, routing algorithm, or link direction (i.e. unidirectional, bidirectional) [KS83].

In practice, all the switches are identical, thus amortizing the design cost. When switches have the same number of input and output ports, MINs also have the same number of input and output ports. Since there is a one–to–one correspondence between inputs and outputs, the connections between stages are also called permutations.

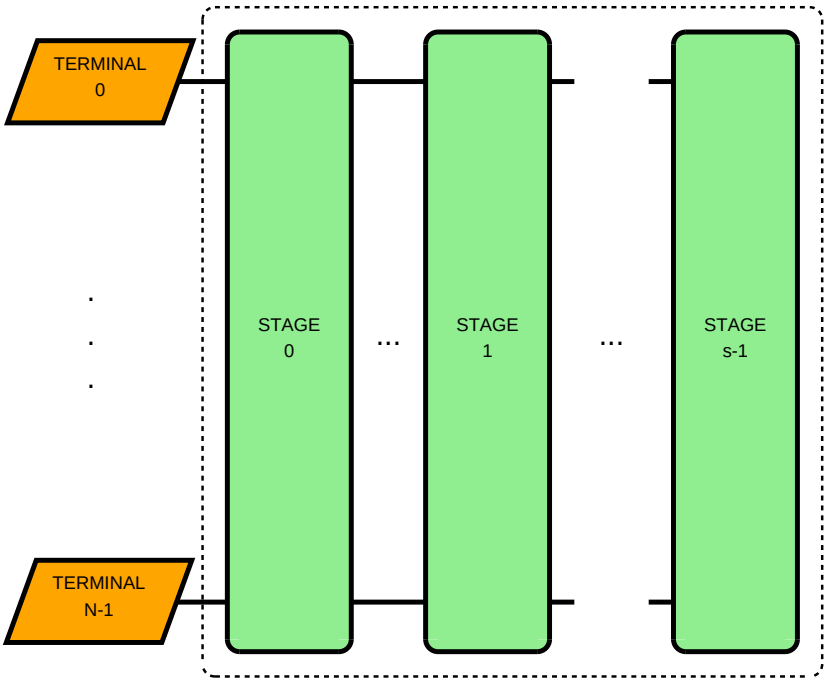


Figure 14: A multistage network with N terminals and s stages.

Depending on the availability of paths to establish new connections, MINs have been traditionally divided into three classes [DYN03]:

1. *Blocking.* A connection between a free input/output pair is not always possible because of conflicts with the existing connections. Typically, there is a unique path between every input/output pair, thus minimizing the number of switches and stages. However, it is also possible to provide multiple paths to reduce conflicts and increase fault tolerance. These MINs have been often implemented because of its simple design and easy control. These blocking networks are also known as multipath networks. Omega network is an example of blocking network (Figure 15a).

2. *Nonblocking.* Any input port can be connected to any free output port without affecting the existing connections. Nonblocking networks have the same functionality as a crossbar. They require multiple paths between every input and output, which in turn leads to extra stages. Therefore, all permutations are supported. However, they are expensive and some require more complex control logic. The best–known example is the Clos network (Figure 15b).

3. *Rearrangeable.* Similarly to nonblocking networks, any input port can be connected to any free output port. However, the existing connections may require rearrangement of paths. These networks also require multiple paths between every input and output, but the number of paths and the cost are smaller than in the case of nonblocking networks. The best-known example of a rearrangeable network is the Beneš network (Figure 15c).

Other aspects that make the difference between MINs are the required number of stages and the permutation used to connect two adjacent stages. Given a connection pattern, the number of stages depends on the number of switch ports. Some commonly known permutations are perfect–shuffle, bit–reversal or butterfly (Section A.3).

Many of the known MINs, such as Omega, flip, cube, butterfly, and baseline, belong to the class of Delta networks [Pat81] and have been shown to be topologically and functionally equivalent [lWF80]. A good survey of those MINs can be found in [aa89].

A.2 Preliminary definitions

In this section some basic concepts of MINs are introduced to make easier the description below. The notation used is based on that used in other studies about these networks, and it aims to provide thoroughness to the study to be held later.

In what follows we take into account the following considerations:

- The input/output terminals are the computing nodes.
- All the switches have the same number of ports.

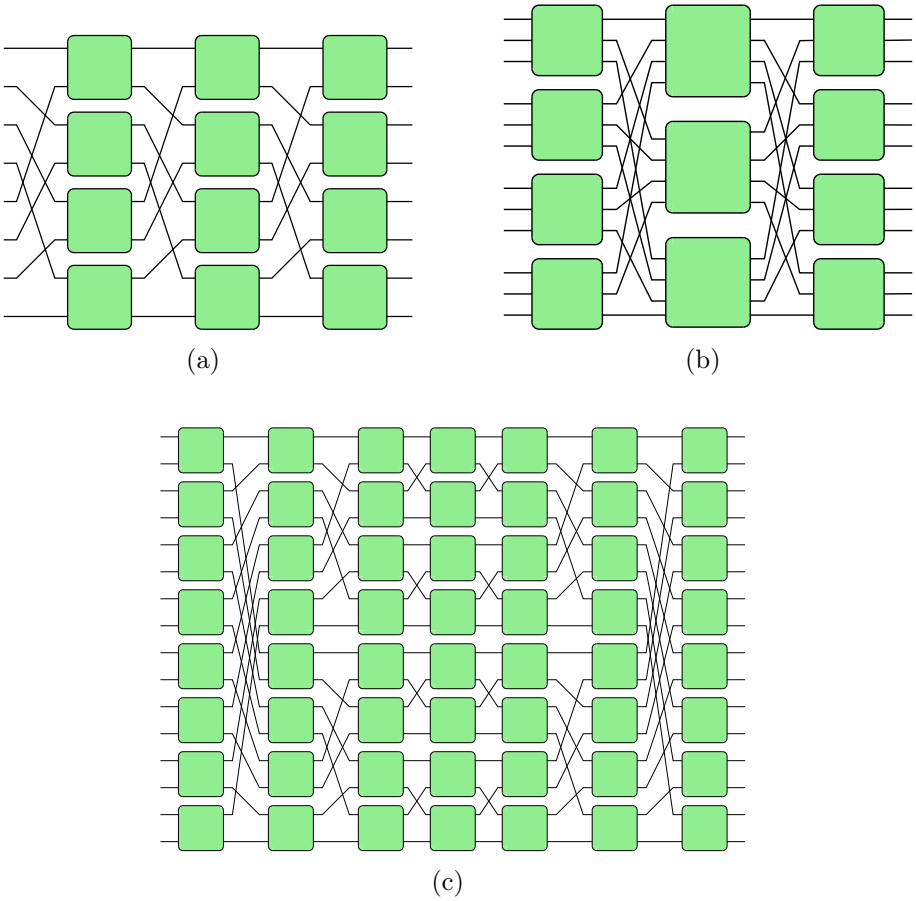


Figure 15: MINs: (a) Omega 8×8, (b) 3–stage Clos, (c) Beněs 8×8.

A.2.1 Notation

We have assumed the following notation (Figure 16):

- N is the total number of terminals (or processing nodes).
- k is the switch arity, or number of ports that connect to terminals/switches in the previous stage and switches in the next stage (if available). Hence, the total number of ports of a $k\times k$ switch is $2k$. The ports faced to the previous stage are numbered from 0 to $k-1$, and the ports connected with the switches in the next stage are labeled from k to $2k-1$.
- Every switch port has an associated global identifier inside the stage, $L = l_{n-1}\dots l_0, 0 \leq l_i < n$, apart from the internal identifier inside the switch. Both identifiers are related by the connection pattern between stages.
- n is the total number of stages, where $n = \log_k N$.
- h is the terminal identifier ($0 \leq h < N$). It consists of a string of n digits $(h_{n-1}\dots h_1 h_0)$, $0 \leq h_i < k$. $\mathcal{H}$ is the set whose members are the terminals of the MIN, verifying card($\mathcal{H}$)=N, where card is the cardinality of sets.

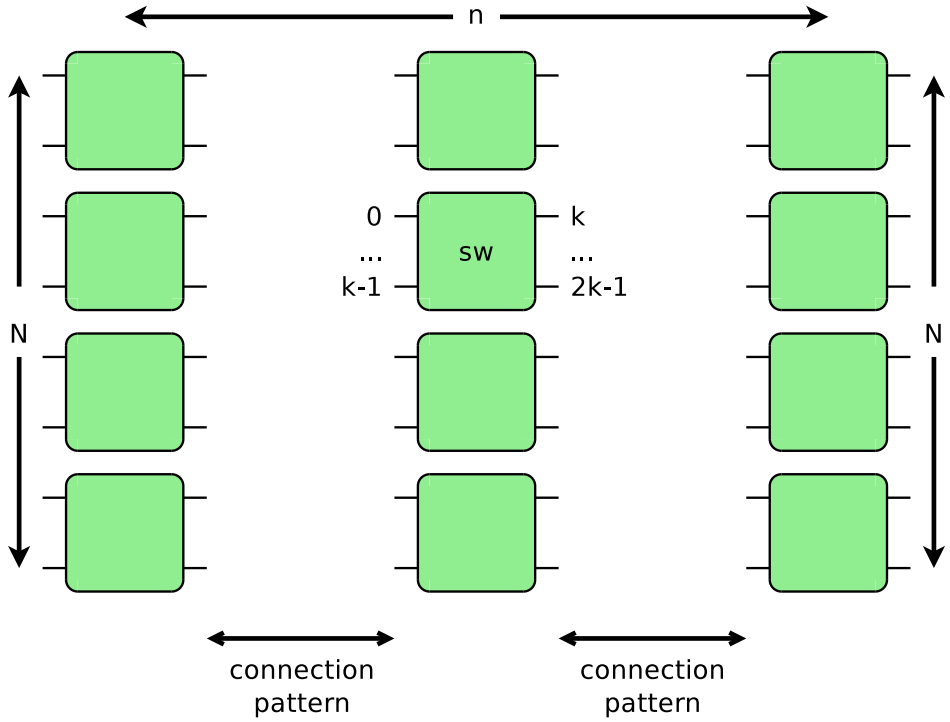


Figure 16: Assumed notation for a MIN with N terminals and $k \times k$ switches.

- $\langle s, o \rangle$ is a tuple that identifies uniquely a switch, where s refers to the stage ($0 \leq s < n$), and $o = o_{n-2}, \ldots, o_1, o_0$ indicates the position of the switch inside the stage, where $0 \leq o_i < k$ and $0 \leq i < n - 1$.

A.3 Connection pattern

Several connection patterns have been proposed to interconnect two adjacent stages of the MIN and the processing nodes with the first and/or the last stage. These patterns correspond to certain well–known permutations. Some of those permutations are defined below.

- **Perfect shuffle**

  The *perfect k–shuffle* permutation σ^k is defined by

  $$\sigma^k(x_{n-1}x_{n-2}\ldots x_1x_0) = x_{n-2}x_{n-3}\ldots x_1x_0x_{n-1}$$

  It performs a cyclic shifting of the digits x_i, $0 \leq i < n$, to the left for one position. Every x_i consists of k bits.

  The *inverse perfect shuffle* does the opposite to the *perfect shuffle* permutation,

  $$\sigma^{k^{-1}}(x_{n-1}x_{n-2}\ldots x_1x_0) = x_0x_{n-1}\ldots x_1$$

- **Digit reversal**

  The *digit reversal* permutation ρ^k is defined by

  $$\rho^k(x_{n-1}x_{n-2}\dots x_1x_0) = x_0x_1\dots x_{n-2}x_{n-1}$$

  This permutation is usually referred to as *bit reversal*. It performs a swapping between digits x_i and x_{n-1-i} where $0 \leq i < n$. Every x_i consists of k bits.

- **Butterfly**

  The ith k–ary *butterfly* permutation β_i^k where $0 \leq i < n$, is defined by

  $$\beta_i^k(x_{n-1}x_{n-2}\dots x_1x_0) = x_{n-1}\dots x_{i+1}x_0x_{i-1}\dots x_1x_i$$

  It interchanges the least significant digit with the ith digit. Every x_i consists of k bits. Note that β_0^k is also called identity permutation.

Additionally, the connection pattern is used to name the MINs. So, we will referred to butterfly MINs, perfect–shuffle MINs and bit-reversal MINs using butterfly, perfect–shuffle and bit–reversal permutations, respectively.

A.4 Unidirectional MINs

In an unidirectional MIN, the channels and switches are unidirectional. Half the terminals are located at one side, and the another half at the opposite side. The MIN is in the middle. Communication is allowed only in one direction. Figure 17 shows an unidirectional switch with k input ports and k output ports.

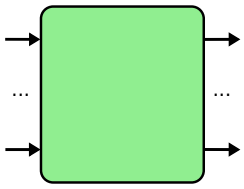


Figure 17: Unidirectional switch.

All paths in an unidirectional MIN go across all the stages. So all paths have the same length. Figure 18 illustrates the unidirectional MIN topology for $N = 8$ nodes built with 2×2 switches.

Banyan networks are a class of MINs with the property that there is a unique path between any pair of source and destination [GL73]. A Delta network is a subclass of banyan networks, which is constructed from identical $k \times k$ switches in n stages, where each stage contains N/k switches. Many of the known MINs, such as Omega, flip, cube, butterfly, and baseline, belong to the class of Delta networks [Pat81] and have been shown to be topologically and functionally equivalent [lWF80]. A good survey of those MINs can be found in [aa89].

The most popular routing algorithm in unidirectional MINs is self–routing.

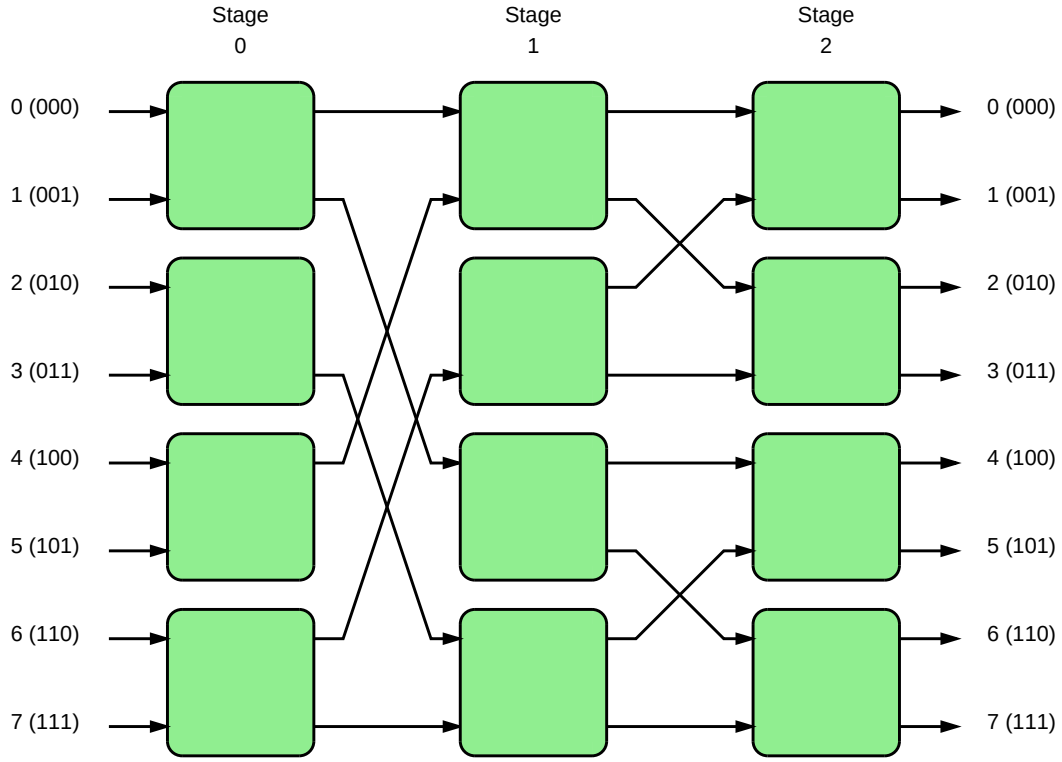


Figure 18: Unidirectional MIN built with 2×2 switches.

A.4.1 Self–routing algorithm

The self–routing algorithm is a deterministic routing algorithm for MIN networks. The routing decision is based on the destination address. The paths are established in a distributed way by using routing tags [Pat81].

The routing function determines the output port taking into account which digit is the least significant one at the i^{th} stage. In unidirectional MINs with $k \times k$ switches, if the value of the digit is i $(0 \leq i < k)$, the packet will be forwarded by the output port $k + i$.

For a $k \times k$ switch, there are k output ports. If the value of the corresponding routing tag is i $(0 \leq i < k)$, the corresponding packet will be forwarded via port i. For an n–stage MIN, the routing tag is $T = t_{n-1} \ldots t_1 t_0$, where t_i controls the switch at the i^{th} stage.

Figure 19 shows the paths followed by one packet from node 0010 to node 1010, and another one from node 0111 to node 1101 in a $N = 16$ butterfly unidirectional MIN with 16×16 switches. The routing tags are 0101 and 1011, respectively. In the first case, the packet is forwarded by the output port 3 in the stages s_0 and s_2. However, it is routed by the output port 2 in the stages s_1 and s_3.

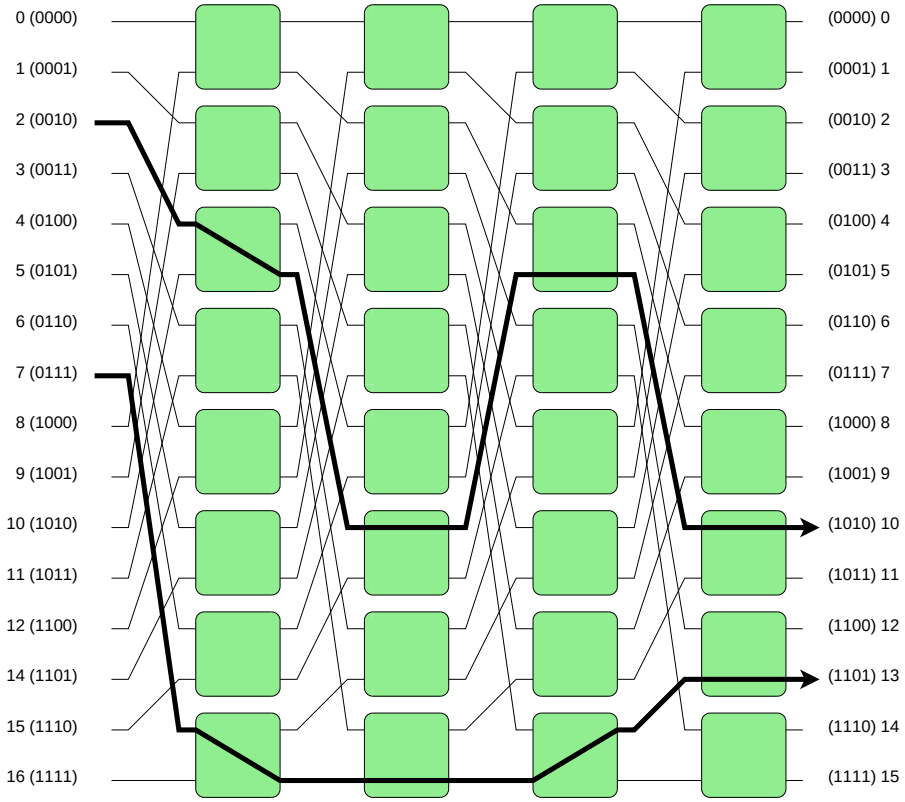
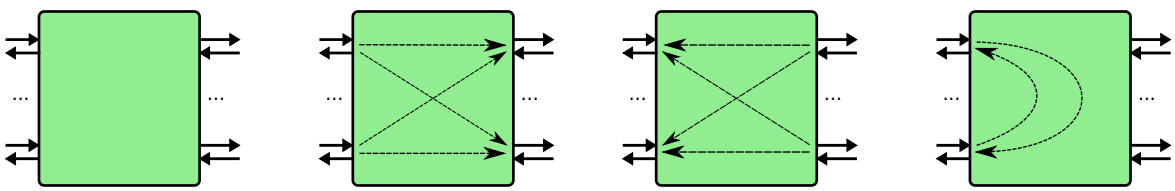


Figure 19: Paths selected by the self–routing algorithm in a $N = 16$ butterfly MIN.

A.5 Bidirectional MINs

Bidirectional MINs (BMINs) are composed of bidirectional switches and switch ports are connected by bidirectional channels (Figure 20(a)). This means that packets can be transmitted simultaneously in opposite directions between neighboring switches. A bidirectional channel is built by joining two unidirectional channels, in opposite directions. In such a way, the bidirectional channel can transmit two packets at the same time in opposite directions between neighboring switches. The bidirectional switch performs three types of internal connections: forward, turnaround and backward connections. Figures 20(b,c,d)) depict the three possible internal connections. As turnaround connections between ports at the same side of a switch are possible, paths have different lengths.



(a) Bidirectional switch. (b) Forward connection. (c) Backward connection. (d) Turnaround connection.

Figure 20: Internal connections in a bidirectional switch.

The network terminals are not directly connected to the switches placed at the stage $j = \log_k N - 1$. Figure 21 shows a $N = 8$ BMIN built with 2×2 switches. Notice that switches of the last stage are not connected to the terminals as it occurs in unidirectional MINs.

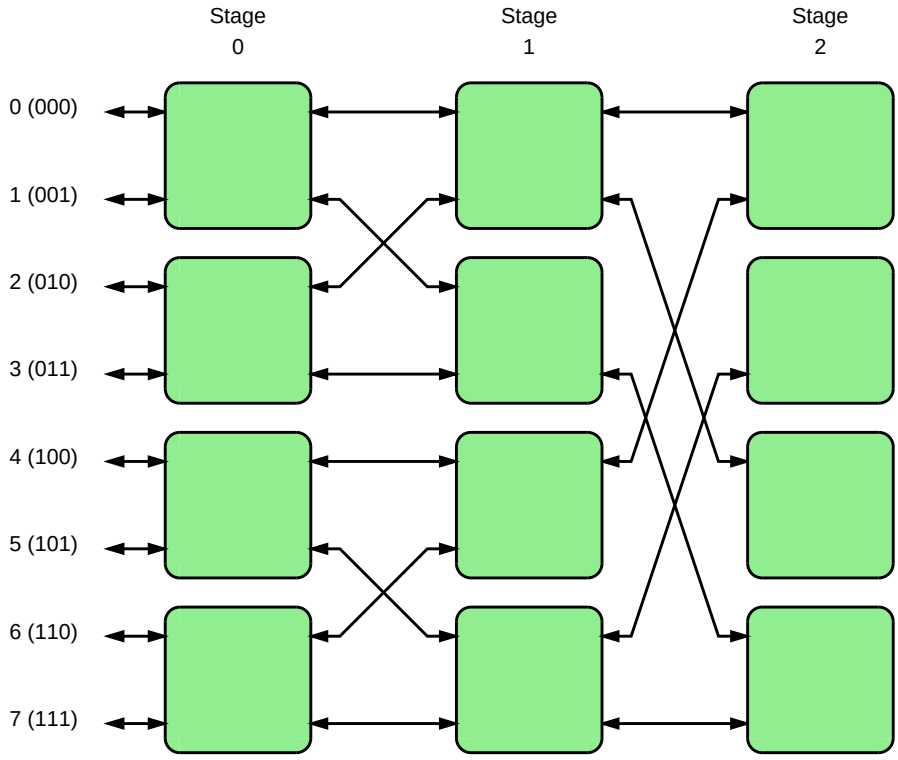


Figure 21: BMIN de $N = 8$ y $k = 2$.

Paths are established in BMINs by crossing stages in the forward direction, then establishing a turnaround connection, and finally crossing stages in the backward direction to the destination terminal. This is usually referred to as turnaround routing (Section A.5.1).

In a BMIN, the routing algorithm could select several minimal paths to send a packet from the source node h to the destination node h'. Firstly, a path goes across stages in the forward direction. Each switch can select any of its k output ports. However, once the turnaround connection is crossed, a single path is available up to the destination node.

According to the definition of the turnaround routing algorithm in the Section A.5.1, the turnaround connection is done in any switch at the stage t ($t = \text{FirstDifference}(h, h')$). So, there are k^t switches where all the possible paths between h and h' could do the turnaround connection.

Turnaround routing algorithm is adaptive in the forward phase because it selects any subpath between the source node h and the k^t switches placed at the stage t. This feature balances the load in the network, and gives fault tolerant characteristics.

On the other hand, after doing the turnaround connection, there is an unique path to the destination node h'. For this reason, the turnaround routing algorithm is deterministic in the descending, or backward, phase, like self–routing is. If no fault tolerance policy is defined, when a switch/channel fails, then the packets could not reach the destination node h'. In [SJS08], the authors propose a strategy to provide fault tolerance in the descending phase in fat–trees.

A butterfly BMIN with turnaround routing can be viewed as a fat tree [Lei85]. In a fat tree, processors are located at leaves, and internal vertices are switches (Section A.6).

A.5.1 Turnaround–routing algorithm

To send a packet from a source node h to a destination node h', it is first sent forward to the least common ancestor of both nodes. Then, the packet is turnaround at stage t (the concrete switch does not matter), and it is sent backward to the destination.

The existing path between a source–destination pair that is obtained by the turnaround routing algorithm can be formalized as follows [NGM97]:

Definition 1.23 *A turnaround routing path between any source and destination pair must meet the following conditions:*

- *The path consists of a sequence of forward connections, one turnaround connection in the stage s, and backward connections.*
- *The number of forward connections is equal to the number of backward connections.*
- *To prevents redundant communication from occurring, no connection is allowed to use the same switch port as input and output port.*

Definition 1.24 *Given $h = h_{n-1}h_{n-2}\dots h_0$ and $h' = h'_{n-1}h'_{n-2}\dots h'_0$ two nodes, the FirstDifference(h,h′) function returns the stage s, ($0 \leq s < n$), where the turnaround connection occurs.*

$$\textit{FirstDifference}(h, h') = s \text{ if and only if } h_s \neq h'_s \text{ and } h_j = h'_j,\ \forall j \in [s+1, n-1]$$

Note, the FirstDifference(h,h') function returns the position where the first (leftmost) different digit appears between h and h'.

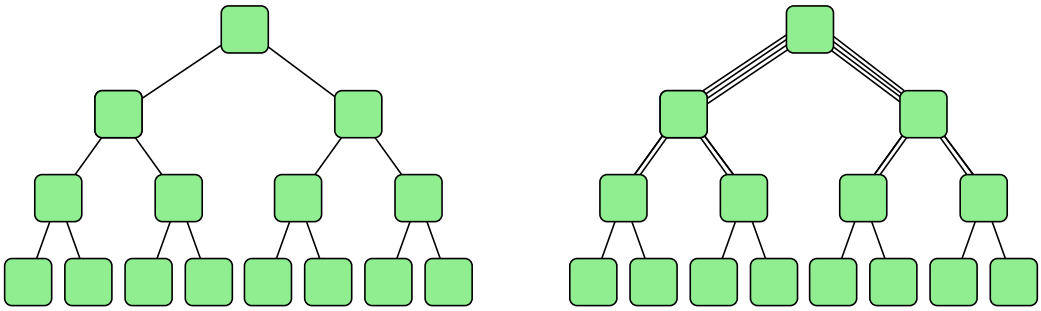
The turnaround routing is deadlock free and shortest path routing [NGM97]. As mentioned, in a BMIN there are multiple choices of the shortest path, which the

turnaround routing may select, between a source and a destination. Specifically, if the turnaround connection is done at the stage s, there are k^s valid paths of minimum length. After the packet is turnaround, there is only path to destination, for this reason, the turnaround routing is said to be adaptive and deterministic in the forward and backward direction, respectively. This feature makes possible to have load–balanced and to design fault tolerant networks.

A.6 *Fat–tree* topology

A *fat–tree* is an indirect interconnection network based on a complete binary tree. Unlike the traditional notion of a tree, where all branches are similar, fat–trees are more like real trees in that they get thicker closer to the root [Lei85].

A binary tree retains the capacity of their channels while fat–tree increases the capacity of the channels as it approaches the root (Figure 22). The processing nodes are located at the leaves of the fat–tree. Each node of the fat–tree corresponds to a switch. Going upwards in the fat–tree, the channels capability increases, but complexity and hardware cost do proportionally. The capacity is determined by the amount of available hardware. This means that a fat–tree topology is also parameterizable in the channel bandwidth.



(a) Binary tree. (b) Fat–tree.

Figure 22: Binary tree and fat-tree.

Routing packets in a fat–tree is quite easy, since there is a unique minimum path between every pair of computing nodes. A message going from node h to node h' goes up the tree to their least common ancestor and then back down according to the least significant bits of h'. Note that at any node of the tree, there are several choices for the routing of a packet. In such cases, the routing algorithm may select one of the channels to distribute the load minimizing the network congestion.

Fat–trees have many desirable properties. The universality theorem proposed by Leiserson states that for any given amount of communications hardware, a fat–tree build from that amount of hardware can simulate every other network built from the same amount of hardware, using only slightly more time (a polylogarithmic factor greater) [Lei85].

The number of switch ports (or switch radix) in the fat–tree increases as going up the tree to the root. This makes unfeasible the physical implementation of the switches. For this reason, Leiserson proposed alternative implementations using switches of fixed radix [LM88]. In Figure 23, the organization of a fat–tree is showed. It is possible to note how the channel capability increases further from the leaves.

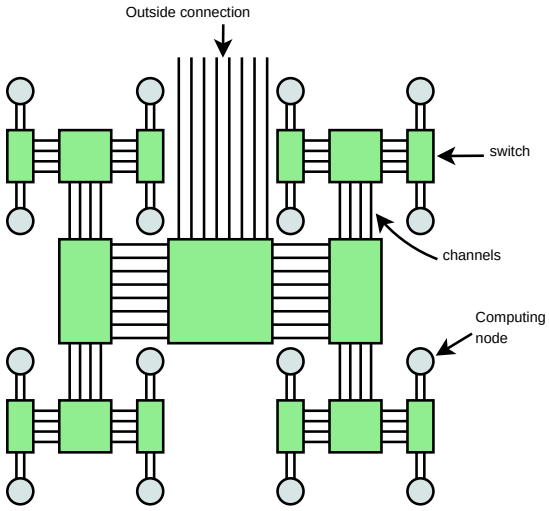


Figure 23: Organization of a fat–tree.

DeHon [DeH90] studied the implementation limitations such wiring, packing complexity and fault tolerant schemes.

The fat–trees are currently the preferred topology for supercomputers [3] like: TianHe-1A at NSC (China), Roadrunner at LANL (USA), and JUGENE at FZJ (Germany), among others.

A.6.1 k–ary n–tree topology

The k–ary n–tree network topology belongs to the family of fat–trees and it is derived from a concrete class of MINs: the k–ary n–butterflies (or k–ary n-flies) [Lei92]. A k–ary n–fly MIN is obtained by applying the β_i^k permutation, $0 \leq i < n$, to obtain the connection patterns between stages.

The k–ary n–tree connect N nodes using nk^{n-1} switches. Two switches $\langle s, o_{n-2} \dots o_0 \rangle$ and $\langle s', o'_{n-2} \dots o'_0 \rangle$ are connected with a channel if $s' = s + 1$ and $o_i = o'_i \ \ \forall\ i \neq s$. Moreover, there is a channel between the switch $\langle 0, o_{n-2} \dots o_0 \rangle$ and the processing node $h = h_{n-1} \dots h_0$ if $o_i = h_{i+1}, 0 \leq i < n-1$.

Figure 24 illustrates a 2–ary 4–tree MIN network for 16 processing nodes with 4 stages and 2×2 switches.

[3] According to the November 2010 Top500 Supercomputing list at www.top500.org.

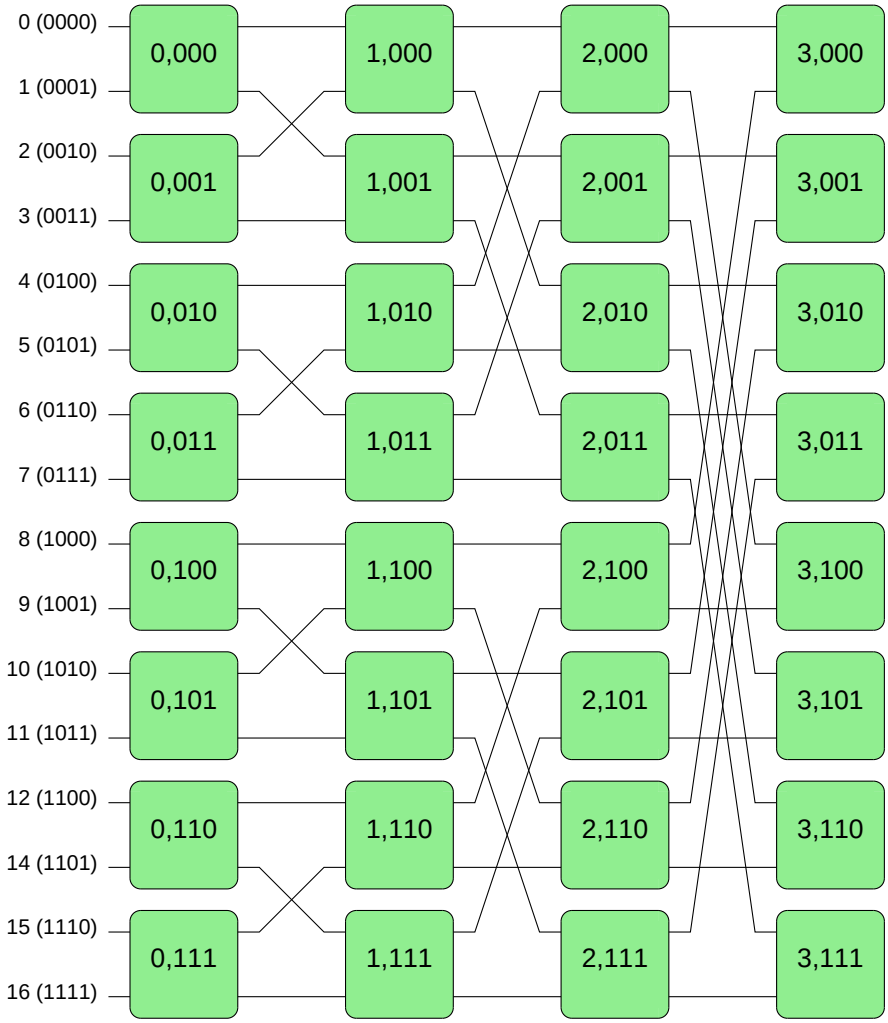


Figure 24: 2–ary 4–tree MIN network.

A.7 Load–balanced routing algorithm

The routing algorithm is the mechanism that determines the path that a message follows on the network to reach its destination, from its source node. Usually, there are multiple paths that can carry a message to its destination; among these paths we can find a set of minimal length paths. A good routing strategy seems to be the one that just uses the minimal paths. There are also other aspects that are usually taken into account when designing routing algorithms, which we have already mentioned led to complex and sophisticated design methodologies for these algorithms.

Taking into account this, the load–balanced routing algorithm concept is introduced. Based on this definition, it would be possible to identify which routing algorithms distribute the generated paths between the network elements in a balanced way.

Definition 1.25 *Given an interconection network $I = G(C, N)$, the routing algorithm R is said to be balanced, or R fully distributes the paths generated in I, if all the switch*

channels in C belonging to I are crossed by the same number of paths.

Definition 1.26 *Given a multistage interconnection network I, the routing algorithm R is said to be balanced, or R fully distributes the paths generated in I, if all the channels of the switches belonging to a given stage, are crossed by the same number of paths.*

For MIN networks there are adaptive routing algorithms that balance partially or fully the paths generated. However, adaptive algorithms arise some difficulties (e.g. out–of–order delivery, more complex implementation) that make deterministic routing algorithms be interesting for some applications.

It is possible to design deterministic routing algorithms, which would balance the paths generated. A simple strategy consists in assigning the output ports in each switch to the difference paths passing through the switches by means of a function that spreads the paths (i.e. load network) between their ports.

For example, let us suppose a $k \times k$ switch that belongs to a concrete stage in a BMIN. From that switch it would be possible to arrive at m destination nodes: $d_0, d_1, \ldots, d_{m-1}$ (m is a multiple of k). The following functions would distribute the destinations between the k output ports of the switch:

- allocation of consecutive destinations to the same port. Let $m' = m/k$ be the number of destination nodes assigned to each output port. Hence,

  To the port l_0 it assigns destinations $d_0, d_1, \ldots, d_{m'-1}$

  To the port l_1 it assigns destinations $d_{m'}, d_{m'+1}, \ldots, d_{2m'-1}$

  ...

  To the port l_{k-1} it assigns destinations $d_{(k-1)m'}, d_{(k-1)m'+1}, \ldots, d_{km'-1}$

- cyclic allocation of consecutive destinations to consecutive ports. Let $m' = m/k$ be the number of destination nodes assigned to each output port. Hence,

  To the port l_0 it assigns destinations $d_0, d_k, \ldots, d_{(m'-1)k}$

  To the port l_1 it assigns destinations $d_1, d_{k+1}, \ldots, d_{(m'-1)k+1}$

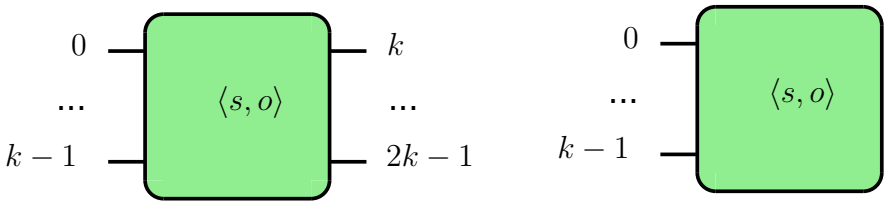
  ...

  To the port l_{k-1} it assigns destinations $d_{k-1}, d_{2k-1}, \ldots, d_{m'k-1}$

The second one, for example, corresponds to the proposal in [GGG+07] and it is defined in the next section.

A.7.1 DESTRO routing algorithm

As described in detail in Section A.2.1, and adapted to BMINs, the $2k$ ports of the switch $\langle s, o\rangle$ are distributed in two disjoint groups. On one hand, the k ports that connect to switches that are located in the previous stage $s-1$, where $0 \leq s < n$, are labeled from 0 to $k-1$ (Figures 25(a) and 25(b)). On the other hand, the other k ports that connect to switches that are located in the stage $s+1$, where $0 \leq s < n-1$, are labeled from k to $2k-1$. Switches belonging to the last stage $n-1$ only use half of the ports, and they are labeled from 0 to $k-1$ (Figure 25(b)).



(a) If $0 \leq s < n-1$. (b) If $s = n-1$.

Figure 25: Numbering scheme for switch ports.

In an informal way, the port l is obtained from the k–ary number, h_s, of the destination node. In switches with $2k$ ports, h_s can represent two different port labels, according to the numbering scheme that has been assumed: $l = h_s$, where $0 \leq h_s < k$; and $l = h_s + k$, where $k \leq l < 2k$. To obtain the exact value of l, the routing function simply needs to know the direction of the path/route of a message (i.e. in forward or backward direction). Hence,

$$R(\langle s, o\rangle, h) = \begin{cases} l = h_s + k, & \text{if the message goes forward} \\ l = h_s, & \text{if the message goes backward} \end{cases}$$

In [GGG+07] the authors propose an implementation of the DESTRO deterministic routing algorithm for fat–trees by means of *Flexible Interval Routing*, (FIR) [GLD05]. FIR is an extension of *Interval Routing*, (IR) [ea91]. In IR, every port has two registers (*First Interval* and *Last Interval*) that define the beginning and end of routing interval, respectively. To send a message through a port, the destination address must be inside the routing interval. FIR adds an extra register (*Mask Register*) per port for defining which bits of the destination address must be compared to the routing interval. In order to guarantee deadlock freedom, FIR adds one more extra register (*Routing Restrictions Register*) per port that establishes the priority between several ports.

Proposition 1.31 *If the traffic pattern is uniform, DESTRO routing algorithm is balanced according to the Definition 1.26.*

Proof: When the traffic is uniformly distributed in the network, the probability of sending traffic from the switch $\langle s, o\rangle$ to a destination node h, $h \in \{N_b^R(\langle s, o\rangle) \cup N_f^R(\langle s, o\rangle)\}$ (see Definitions 1.18 and 1.19), is constant.

All the ports l, $0 \leq l < k$, of the k^{n-1} switches at the stage s, $0 \leq s < n$, receive the same number of paths in backward direction according to Definition 1.20, because

$$card(N_b^R(\langle s, o_0\rangle, l_j)) = \cdots = card(N_b^R(\langle s, o_i\rangle, l_k)) = \cdots = card(N_b^R(\langle s, o_{(k^{n-1})}\rangle, l_m))$$
$$\forall l_j, l_k, l_m \in [0, k-1]$$

Similarly, all the ports l', $k \leq l' < 2k$, of the previous k^{n-1} switches, receive the same number of paths in forward direction according to Definition 1.21, because

$$card(N_f^R(\langle s, o_0\rangle, l'_j)) = \cdots = card(N_f^R(\langle s, o_i\rangle, l'_k)) = \cdots = card(N_f^R(\langle s, o_{(k^{n-1})}\rangle, l'_m))$$
$$\forall l'_j, l'_k, l'_m \in [k, k-1]$$

□