*Technical Report CS-93-209, The University of Tennessee, Knoxville, USA, 1993.*

# Efficient Communication Operations in Reconfigurable Parallel Computers

F. Desprez[*], A. Ferreira[†]
CNRS - Laboratoire LIP, CNRS-URA 1398
ENS Lyon
46, Allée d'Italie
69364 LYON Cedex 07
France
email: [desprez,ferreira]@lip.ens-lyon.fr

B. Tourancheau [‡§]
The University of Tennessee
Computer Science Department
107, Ayres Hall
Knoxville, TN 37996-1301
USA
e-mail: btouranc@cs.utk.edu

**Abstract**

Reconfiguration is largely an unexplored property in the context of parallel models of computation. However, it is a powerful concept as far as massively parallel architectures are concerned, because it overcomes the constraints due to the bissection width arising in most of distributed memory machines. In this paper, we show how to use reconfiguration in order to improve communication operations that are widely used in parallel applications. We propose quasi-optimal algorithms for broadcasting, scattering, gossiping and multi-scattering.

**Keywords:** Reconfiguration, broadcast, scattering, gossiping, communications, distributed memory parallel computers

## 1 Introduction

For massively parallel architectures, the hardware complexity of the interconnection network is much higher than that of the processing units: "the interconnection network employs 99% of the hardware involved" [JMM92]. Moreover, due to the communication-intensive nature of most computational tasks, their performance depends heavily on the underlying interconnection network.

Reconfiguration is largely an unexplored property in the context of parallel models of computation. The involved literature is focused on three areas: architectures for parallel image processing, unit time reconfiguring models (having an underlying wired network), and reconfiguration to eliminate hardware faults. Our approach is that reconfiguration may be utilized to enhance communication in general parallel computers. The idea is that when large pools of data are to be exchanged between two sets of processors, some time may be spent in reconfiguring the system to

*Partially supported by MRE grant No. 974, the CNRS-NSF grant No. 950.22/ 07 and the research program C3 of the French CNRS

†Partially supported by ANM and C3, and by the French-Israeli project on Optical Models for Parallel Computing

‡On leave from LIP, CNRS URA 1398, ENS Lyon, 46 allée d'Italie, 69364 Lyon Cedex 07, France.

§This work was supported in part by the National Science Foundation under grant ASC-8715728, the National Science Foundation Science and Technology Center Cooperative Agreement CCR-8809615, the DARPA and ARO under contract DAAL03-91-C-0047, CNRS-NSF grant 950.223/07, Archipel SA and MRE under grant 974, research program PRC $C^3$, and DRET.

create a high bandwidth channel (or a set of direct channels) between them. After reconfiguration time (during which local processing may proceed), the data is transmitted in a very short time. Thus, except for the reconfiguration time, there are no bottlenecks in the interconnection media and the notion of "locality" varies dynamically according to the specific computation requirements.

In this paper we study global communication schemes, commonly used in all areas of parallel computing, ranging from neural networks to linear algebra. We propose a numbering scheme for the processors in order to implement data-communication reconfiguring algorithms, under the linear models of communication and reconfiguration, for the following problems. (Suppose that all messages are of the same size, and let $t_r$ be the reconfiguration time, and $\beta$ and $\tau$ be the communication startup and inverse of the bandwidth times.)

- One-to-all broadcasting: broadcasting from one node to all other nodes.
- One-to-all personalized broadcasting (scattering): one node sends a distinct message to all other nodes.
- All-to-all communication: broadcasting from each node to every other nodes.
- All-to-all personalized communication: each node sends a distinct message to every other node.

We show how to implement the first operation in such a way that there is a machine-dependable tradeoff between $t_r, \beta$, and $\tau$. For all the others, our algorithms come very close to the lower bound for $\beta$ and $\tau$, while keeping the reconfiguration cost as low as possible.

Our paper is organized as follows. In next section a brief overview of reconfiguration is given. Then, the model in which our algorithms are analyzed is well defined in section 3. The numbering scheme for the reconfiguring communication pattern is introduced in section 4, and all the new algorithms are presented and discussed in section 5. We close the paper with a cross-analysis of all the algorithms and ways for further research.

## 2 Reconfiguration

A classification of the reconfiguration capabilities of today's multicomputers is given in [BBM91, BDT93]. Parallel distributed memory machines can be classified in 4 categories, depending on the reconfiguration capabilities of their interconnection network, as follows.

- **static:** the interconnection network is fixed and cannot be modified. This is the case of most architectures.
- **quasi-static:** the interconnection network is set before the execution starts. It remains unchanged during the runtime of the program.
- **quasi-dynamic:** programs are divided into series of phases. Each phase uses its own topology and the interconnection network is set before a phase starts. Within phases, the topology remains fixed.
- **dynamic:** Links are asked to a configuration manager. This model is close to the circuit-switching communication protocol, but for the bissection width constraints.

Some projects of reconfigurable architectures using Transputers and their switching chip $C004$ have raised in the middle of the 80'. These architectures depend on which reconfiguration model they implement. In the following we describe some related works concerning reconfiguring parallel computers.

The Parsifal T-Rack [KA91, TM90, JM89] implements a partial reconfiguration. Processors are connected in a ring of 64 nodes and the remaining links can be connected dynamically through a switching network. The DAMP project [BBM91] implements dynamic reconfiguration using a static interconnection network and dynamic connections between transputers in different modules. This is close to circuit switching communication protocol where a path is set before communication occurs. The CNET environment [ABB+91] uses a phase based protocol where topologies are set before the start of the current phase, being therefore quasi-static [BBM91, BDT93]. The target architecture is a SuperNode machine [Wai90].

Concerning optical networks, the current trend is to use free space optics, with reconfiguration, for parallel computing. Designs like the Electro-Optical Crossbar have been proposed [IKYI], that uses a hybrid reconfiguration technique for interconnecting processors. There are $N$ processors, each located in a distinct row and column of the $N \times N$ processing layer. For each processor, there is a hologram module having $N$ units, such that the $i^{th}$ unit has a grating plate with a frequency leading to a deflection angle corresponding to the processor located at the grid point $(i, i)$. In addition, each unit has a simple controller and a laser beam. To establish or reconfigure to a new connection pattern, each processor broadcasts the desired destination processor's address to the controller of each of $N$ units of its hologram module, using an electrical bus. The controller activates a laser (for conversion of the electrical input to optical signal) if its ID matches the broadcast address of the destination processor. The connection is made when the laser beams are passed through the predefined gratings. Therefore, since the grating angles are predefined, the reconfiguration time of this design is bounded by the laser switching time, which is in the order of nano-seconds using Gallium Arsenide (GaAs) technology.

The Milord project from the ONERA/CERT [TCS88] uses optical technology for the switching network of a transputer based machine. It implements phases model of reconfiguration due to the cost of the switching cost. The optical crossbar is based on a matrix-vector concept using Spacial Light Modulator devices. The reconfiguration time is about 200 milliseconds for the whole network.

## 3 Model of analysis

We assume that we have a distributed parallel machine with $P$ processors $(0..P-1)$. Each processor has independent units for communication and computation. Thus, on the same node it is possible to perform simultaneously, bidirectional data transfer on each link (full duplex and $k$-port assumptions [FL91]) and arithmetic operations.

Each node is assumed to have $k$ links $(0..k-1)$ to connect with other nodes. This assumption allows us to realize every topology with a degree less or equal to $k$. We assume that $k \ll P$.

The communication protocol is a "point-to-point rendez-vous". However, with the exception of the communication "handshake", all the processors are independent and run asynchronously.

The time needed to communicate a message of size $L$ between two processors is modeled as a linear function of the start-up time $\beta$, the length $L$ and the propagation time $\tau$ of the communication channel. Here we shall use the communication cost as defined in [SS89]: $(t_{com} = \beta + L\tau)$.

Several models can be used for the reconfiguration cost. We could, for instance, assume that the reconfiguration time is negligible in comparison to communications but unfortunately this is not realistic.

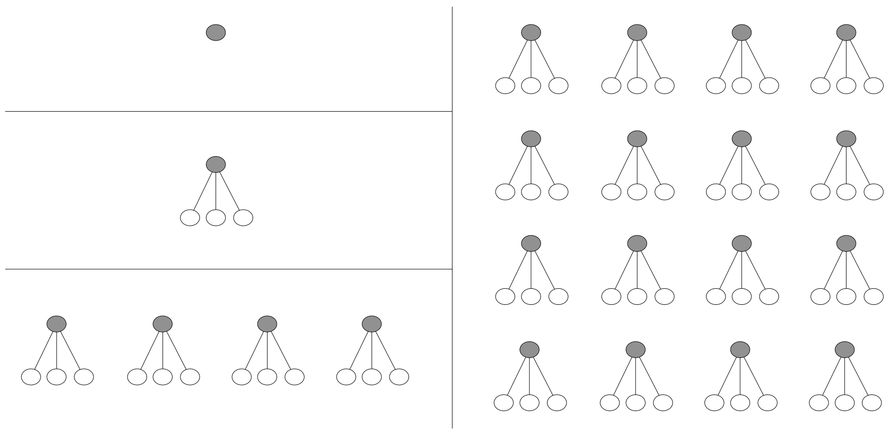

Figure 1: RCP algorithm for 27 nodes and degree 3

Previous experiments with reconfigurable parallel computers [BDT93, DT92] taught us that the reconfiguration cost in existing architectures is best modeled linearly in the number of reconfigured links, i.e. for $n$ links set, the cost of reconfiguration is $t_r = \beta_r + n\tau_r$.

## 4 Reconfigurating Communication Pattern

In this section, we describe the Reconfigurating Communication Pattern (RCP), a numbering scheme that defines an algorithm where, from a special node – supposed hereinafter and without loss of generality to be node 0 –, degree $k$ topologies are built at each step. At its completion, the RCP guarantees that for any node in the system there is a path – in time – connecting node 0 to it. And this, in a minimum number of steps.

In a first step, node 0 is connected to $k$ processors. In the next step, every already reached processor is able to have the same behavior as node 0 in the first step. This is iterated until all the processors involved in the communication operation are reached (see algorithm in Figure 2 and an example for $N = 27$ and $k = 3$ on Figure 1).

Clearly, $\log_{k+1}(N)$ is the number of steps needed to reach every node. Then, there are $\log_{k+1}(N)$ reconfiguration steps. Since the RCP can be seen as a forest of k-ary trees, the total number of modified links is $N - 1$.

Hence, under the model defined in the previous section, the reconfiguration cost is given by:

$$T^{RCP}_{reconf} = \log_{k+1}(N)\beta_r + (N-1)\tau_r. \tag{1}$$

Let $l$ be the current step and $i$ be the number of a node already reached. Therefore, we define the RCP in such a way that node $i$ in step $l$ will be connected to nodes:

$$(k+1)^l + ik,\ (k+1)^l + ik + 1,\ (k+1)^l + ik + 2,\ \cdots \ \text{and}\ \ (k+1)^l + ik + k - 1$$

## 5 Global communication schemes

In the remainder of the paper, let $h = \log_{k+1}(N)$.

```
i = my_node_id
for $l = 0$ to $\log_{k+1}(N)$ do
 if $i < (k+1)^l$ /* I have been reached at step $l-1$ */
  for $j = 0$ to $k-1$ do
   connect $i$ to node $(k+1)^l + ik + j$
  endfor
 else
  if $i < (k+1)^{l+1}$ I'm connected at this step
  else /* I'm not concerned at this step */
  endif
 endif
endfor
```

Figure 2: Algorithm implementing the RCP

## 5.1 RCP Scattering algorithm

The scattering operation, or personalized one-to-all, is used for instance in the distribution of data on a set of processors [BDT93, Ede91, JH89].

We assume that node 0 owns $N$ packets of size $L$ that have to be delivered one to each node in the system. In the first step of the algorithm, node 0 divide the set of data in $k+1$ equal parts. One of these packets remains on the node (it contains the data used in the next steps), while the $k$ others are sent to the $k$ nodes connected at this step.

At a step $l$, when the communication has complete, reconfiguration occurs and each processor having been reached in a previous step behaves as node 0 did at the first step. Thus, there are $k+1$ nodes having a message of length $\frac{NL}{(k+1)^l}$ bytes. All the reached nodes are then connected to $k$ "sons" and send one $(k+1)$-th of the message to each one.

The algorithm stops when the message at every node is of length $L$ (the size of the personal message). It is not difficult to see that, at this point, all the nodes have been reached.

The reconfiguration cost is given by the reconfiguration cost of the RCP:

$$\begin{aligned} T_{rec}^{scat} &= T_{rec}^{RCP} \\ &= \log_{k+1}(N)\beta_r + (N-1)\tau_r. \end{aligned} \tag{2}$$

And the total communication cost equals the sum of the costs for each step:

$$\begin{aligned} T_{com}^{scat} &= \sum_{i=1}^{h}\left(\beta + \frac{NL}{(k+1)^i}\tau\right) \\ &= \log_{k+1}(N)\beta + \frac{N-1}{k}L\tau. \end{aligned} \tag{3}$$

## 5.2 RCP broadcasting

The broadcast is one of the basic operations on a number of parallel algorithms. It is used for example, to load the same code or data, or to communicate results on a network of processors. This operation has been extensively studied for distributed memory architectures [FL91, JH89, SW87] under different models of communication [FL91].

### 5.2.1 Naïve algorithm

If we imagine that in the RCP, node 0 has a message to be broadcast, then a straightforward broadcast algorithm is obtained, where at each step, every reached node sends the message along its $k$ links.

Since each step costs $(\beta + L\tau)$ in communication time, we have a total cost of:

$$T_{b\ rec} = T^{RCP}_{reconf} + \log_{k+1}(N)(\beta + L\tau) \tag{4}$$

### 5.2.2 Two phase algorithm

Suppose now that we change the naïve algorithm above by splitting the original message in $(k+1)$ parts in the beginning of the algorithm. Then, the naïve algorithm proceeds and once all the processors were reached, node 0 has the whole message and each processor has $1/(k+1)$ of the message. Hence, in order to finish the broadcasting operation, the message must be rebuilt in the processors. For this, complementary processors are connected in $(k+1)$-cliques and they exchange through all their channels the missing parts of the message. Since each $(k+1)$-clique has the whole message, divided among the processors, this single all-to-all operation allows every processor to recover all missing parts of the original message.

Notice that we can recursively apply this idea. We get then a *first* phase, where the message is continuously split according to the RCP; followed by a *second* phase, where the message is step after step completely recomposed, with the help of all-to-all operations implemented in $(k+1)$-cliques of complementary processors.

It is clear that the number of reconfigured links is larger during the second phase because it concerns cliques and no longer trees. Therefore, the reconfiguration cost will be much higher. Actually, the longer the splitting phase, the more expensive the second phase. Then, the savings in the communication time yielded by the splitting disappear under the weight of reconfiguration. One idea is then to stop splitting somewhere in the middle of the first phase, in some step $h'$ $(0 \leq h' \leq \log_{k+1}(N))$. The first phase will thus be composed by a splitting phase followed by a naïve phase, where the naïve broadcasting algorithm is implemented.

An example with $N = 27$ and $k = 3$ is given in figure 3. There are 3 steps in the first phase, but only 2 splitting phases (i.e., $h' = 2$) and thus 2 all-to-all steps in the second phase.

In the following, we give the complexity analysis of the two main phases of this algorithm.

$$T^{phase1}_{com} = \sum_{i=1}^{h'} \left(\beta + \frac{L}{(k+1)^i}\tau\right) + \sum_{i=h'+1}^{h} \left(\beta + \frac{L}{(k+1)^{h'}}\tau\right) \tag{5}$$

$$T^{phase2}_{com} = \sum_{i=h'}^{1} \left(\beta + \frac{L}{(k+1)^i}\tau\right) \tag{6}$$

giving a total communication cost:

$$T^{bcast}_{com} = 2\sum_{i=1}^{h'} \left(\beta + \frac{L}{(k+1)^i}\tau\right) + \sum_{i=h'+1}^{h} \left(\beta + \frac{L}{(k+1)^{h'}}\tau\right)$$

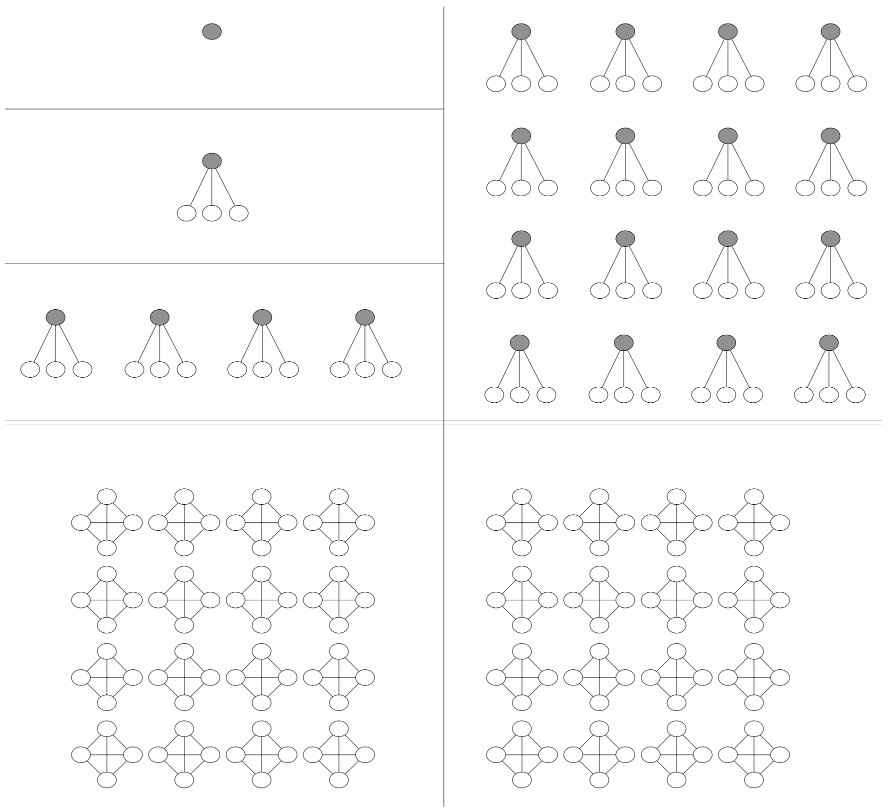

Figure 3: Broadcast algorithm using RCP, for 27 nodes and $k = 3$

$$
\begin{aligned}
&= (h'+h)\beta + \left(\frac{2}{k}\left((k+1)^{h'}-1\right)+h-h'\right)\frac{L\tau}{(k+1)^{h'}} \\
&= (h'+\log_{k+1}(N))\,\beta + \left(\frac{2}{k}\left((k+1)^{h'}-1\right)+\log_{k+1}(N)-h'\right)\frac{L\tau}{(k+1)^{h'}} \\
&\leq (h'+\log_{k+1}(N))\,\beta + \frac{2}{k}L\tau + \frac{(\log_{k+1}(N)-h')}{(k+1)^{h'}}L\tau \qquad (7)
\end{aligned}
$$

The reconfiguration cost is given by:

$$
\begin{aligned}
T^{scat}_{phase1} &= T^{RCP}_{rec} \\
&= \log_{k+1}(N)\beta_r + (N-1)\,\tau_r \qquad (8) \\
T^{phase2}_{reconf} &= h'\left(\beta_r + \frac{Nk}{2}\tau_r\right), \qquad (9)
\end{aligned}
$$

giving a total reconfiguration cost:

$$
T^{bcast}_{reconf} = (h'+\log_{k+1}(N))\beta_r + \left[(N-1)+h'\frac{Nk}{2}\right]\tau_r. \qquad (10)
$$

If we assume that the splitting-phase lasts until the end of the RCP, i.e., $h'=h$, the communication cost is given by:

$$
\begin{aligned}
T^{bcast}_{com} &= 2\left(\log_{k+1}(N)\beta + \frac{N-1}{kN}L\tau\right) \\
&\leq 2\log_{k+1}(N)\beta + \frac{2L\tau}{k}, \qquad (11)
\end{aligned}
$$

and the reconfiguration cost is given by:

$$
T^{bcast}_{reconf} = 2\log_{k+1}(N)\beta_r + \left[(N-1)+\log_{k+1}(N)\frac{Nk}{2}\right]\tau_r, \qquad (12)
$$

The total cost of the RCP broadcasting $T^{bcast}$ is given by the sum of the reconfiguration cost $T^{bcast}_{reconf}$ and the communication cost $T^{bcast}_{com}$.

Now we want to compute the optimal number of splitting steps ($h'_{opt}$) which minimizes the total execution cost. To obtain it, we derive the total cost function with regard to $h'$.

We first assume that the reconfiguration cost is zero.

$$
\begin{aligned}
\frac{\partial T^{bcast}_{com}}{\partial h'} &= 0 \\
h' &= \frac{\log(N)k+k-2\log(k+1) - O\left(\frac{e^{\frac{\log(N)k+k-2\log(k+1)}{k}}\beta}{L\tau}\right)k}{k\log(k+1)}. \qquad (13)
\end{aligned}
$$

If we take into account the reconfiguration cost in our minimization, we have:

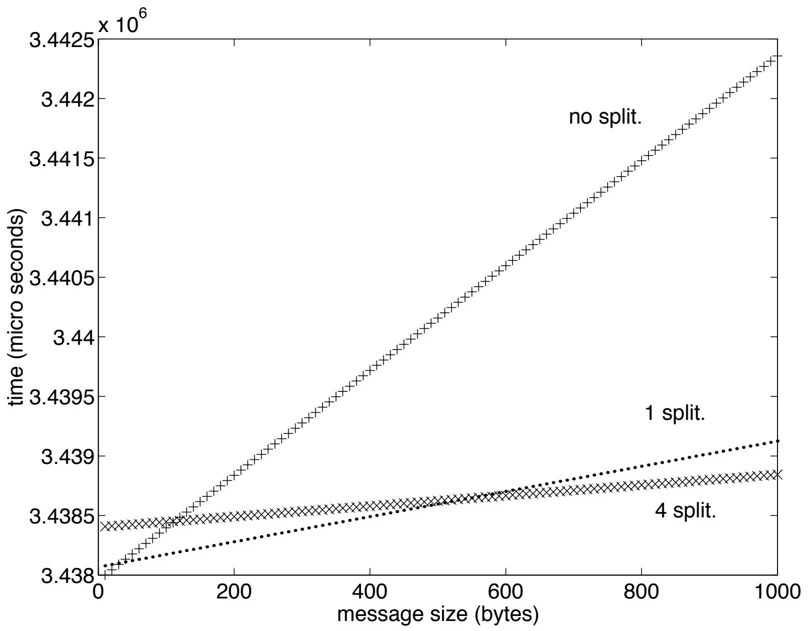

Figure 4: RCP broadcasting on 3125 nodes for $k = 4$

$$\begin{aligned}
\frac{\partial(T^{bcast}_{com} + T^{bcast}_{reconf})}{\partial h'} &= 0 \\
h' &= \frac{\log(N)k + k - 2\log(k+1) - O\left(\frac{e^{\frac{\log(N)k+k-2\log(k+1)}{k}}(2\beta+2\beta_r+Nk\tau_r)}{2L\tau}\right)k}{k\log(k+1)} \qquad (14)
\end{aligned}$$

Roughly, for the communication part, there is a tradeoff between $\log_{k+1}(N)\beta$ and $2\log_{k+1}(N)\beta$ for the startup cost and between $\log_{k+1}(N)\tau$ and $\frac{2L\tau}{k}$ for the propagation time, as follow.

| h' | Reconfiguration | Communication |
|---|---|---|
| 0 | $\log_{k+1}(N)\beta_r + (N-1)\tau_r$ | $\log_{k+1}(N)(\beta + L\tau)$ |
| $\log_{k+1}(N)$ | $2\log_{k+1}(N)\beta_r + \left((N-1) + \log_{k+1}(N)\frac{Nk}{2}\right)\tau_r$ | $2\log_{k+1}(N)\beta + \frac{2L\tau}{k}$ |

Figure 4 shows the time ellapsed for the RCP broadcasting of messages from 10 to 1000 bytes on 3125 nodes in a theoretical degree 4 target machine, with the following parameters. We let $h'$ equals 0, 1 and 4 (recall that $\log_5(3125) = 5$). Notice that, since $\tau_r$ is the cost of a memory allocation, it is negligible with regard to $\beta_r$.

| $\beta_r$ | $\beta$ | $\tau$ |
|---|---|---|
| 100 $\mu$s | 11.5 $\mu$s | 0.88 $\mu$s/byte |

Notice that for small messages ($\leq$ 20 bytes) the splitting phase is not recommended.

## 5.3 All-to-all algorithm

The all-to-all operation is usually very costly and is used in many parallel algorithms when all the processors have to exchange their data [JH87]. This operation is sometimes called gossiping [FL91] or complete broadcast [Sei89].

This operation can be efficiently implemented on a reconfigurable network using several steps of all-to-all operations on $K_{k+1}$ topologies. The number of steps remains the same as for the one-to-all type operations (broadcast and scattering).

The reconfiguration cost is given by:

$$\begin{aligned} T^{ata}_{reconf} &= \sum_{i=0}^{h-1} \left(\beta_r + \frac{Nk}{2}\tau_r\right), \\ &= h\left(\beta_r + \frac{Nk}{2}\tau_r\right) \\ &= \log_{k+1}(N)\left(\beta_r + \frac{Nk}{2}\tau_r\right) \end{aligned} \tag{15}$$

and the communication cost is given by:

$$\begin{aligned} T^{ata}_{com} &= \sum_{i=0}^{h-1} \left(\beta + (k+1)^i L\tau\right). \\ &= h\beta + \frac{(k+1)^h - 1}{k} L\tau \\ &= \log_{k+1}(N)\beta + \frac{(N-1)L}{k}\tau \end{aligned} \tag{16}$$

## 5.4 Personalized all-to-all algorithm

This operation, also called total exchange [Ede91] or multi-scattering [FL91], is used, for instance, in the transposition of a matrix stored by rows or columns [JH87].

We use the same reconfiguration method as for the all-to-all scheme, and at each step of the algorithm, each node sends a message of size $\frac{NL}{k+1}$.

$$\begin{aligned} T^{pata}_{reconf} &= T^{ata}_{reconf} \\ T^{pata}_{com} &= \sum_{i=0}^{h-1} \left(\beta + \frac{NL\tau}{k+1}\right) \\ &= \log_{k+1}(N)\left(\beta + \frac{NL}{k+1}\tau\right) \end{aligned} \tag{17}$$

# 6 Summary

In tables 1 and 2, we give a summary of the complexity results obtained. One can see that our algorithms match the lower bound for three of the problems. As far as broadcasting is concerned, we recall that there is a tradeoff between the factor on $\beta$ and the one on $\tau$, depending on the length

| Algorithm | Communication cost | Lower bound |
|---|---|---|
| ota ($h' = 0$) | $\log_{k+1}(N)(\beta + L\tau)$ | $\max\left(\log_{k+1}(N)\beta, \frac{L}{k}\tau\right)$ |
| ota ($h' = h$) | $2\log_{k+1}(N)\beta + \frac{2L\tau}{k}$ | $\max\left(\log_{k+1}(N)\beta, \frac{L}{k}\tau\right)$ |
| pota | $\log_{k+1}(N)\beta + \frac{(N-1)L}{k}\tau$ | $\max\left(\log_{k+1}(N)\beta, \frac{(N-1)L}{k}\tau\right)$ |
| ata | $\log_{k+1}(N)\beta + \frac{(N-1)L}{k}\tau$ | $\max\left(\log_{k+1}(N)\beta, \frac{(N-1)L}{k}\tau\right)$ |
| pata | $\log_{k+1}(N)\left(\beta + \frac{NL}{k+1}\tau\right)$ | $\max\left(\log_{k+1}(N)\beta, \frac{(N-1)L}{k}\tau\right)$ |

Table 1: Summary of the Communication costs and lower bounds

| algorithm | reconfiguration cost |
|---|---|
| ota ($h' = 0$) | $\log_{k+1}(N)\beta_r + (N-1)\tau_r$ |
| ota ($h' = h$) | $2\log_{k+1}(N)\beta_r + \left((N-1) + \log_{k+1}(N)\frac{Nk}{2}\right)\tau_r$ |
| pota | $\log_{k+1}(N)\beta_r + (N-1)\tau_r$ |
| ata | $\log_{k+1}(N)\left(\beta_r + \frac{Nk}{2}\tau_r\right)$ |
| pata | $\log_{k+1}(N)\left(\beta_r + \frac{Nk}{2}\tau_r\right)$ |

Table 2: Summary of the Reconfiguration costs

of the splitting phase. If we are willing to compromise a factor two in front of $\beta$, then we can save a factor of as much as $k/2$ in front of $\tau$.

We remark that the numbering scheme introduced in Section 4 works only for multi-computers with a number of nodes which is exactly a power of $(k+1)$. Finding a numbering scheme for an arbitrary number of nodes is not difficult if we lose a little of the efficiency of the algorithms. On the other hand, matching the lower bounds in this case seems a very challenging problem.

## References

[ABB$^+$91] J.M. Adamo, J. Bonneville, C. Bonello, P. Moukeli, N. Alhafez, and L. Trejo. Developing a High Level Programming Environment for Supernode. In *Proceedings of Transputer Application - Glasgow, (UK)*, pages 632–637. IOS Press, 1991.

[BBM91] A. Bauch, R. Braam, and E. Maehle. DAMP - A Dynamic Reconfigurable Multiprocessor System with a Distributed Switching Network. In *Proceedings of the 2nd European Distributed Memory Computing Conference - Munich*, Lecture Notes in Computer Science, pages 495–504. Springer Verlag, April 1991.

[BDT93] C. Bonello, F. Desprez, and B. Tourancheau. Parallel BLAS and BLACS for Numerical Algorithms on a Reconfigurable Network. In S. Atkins and A.S. Wagner, editors, *NATUG6 - Transputer Research and Applications 6*, pages 21–38. IOS Press, 1993.

[DT92] F. Desprez and B. Tourancheau. Parallel BLAS and BLACS on a Reconfigurable Network. In M. Garbey and H. Kaper, editors, *NATO Advanced Workshop on Asymptotic-Induced Numerical Methods for Partial Differential Equations, Critical Parameters and Domain Decomposition - Beaune*. NATO ASI Series, 1992.

[Ede91] A. Edelman. Optimal Matrix Transposition and Bit Reversal on Hypercubes: All to All Personalized Communication. *Journal of Parallel and Distributed Computing*, 11:328–331, 1991.

[FL91] P. Fraigniaud and E. Lazard. Methods and Problems of Communication in Usual Networks. Technical Report 91-33, Laboratoire LIP, 1991.

[IKYI] H. Ito, N. Komagata, H. Yamada, and Humio Inaba. New structure of laser diode and light emitting diode based on coaxial transverse junction. Technical report, Research Institute of Electrical Communication, Tohoku University, Sendai 980, Japan.

[JH87] S.L. Johnsson and C.T. Ho. Algorithms for Matrix Transposition on Boolean n-cube Configured Ensemble Architectures. Technical Report YALEU/DCS/TR-572, Department of Computer Science - Yale University, September 1987.

[JH89] S.L. Johnsson and C.T. Ho. Optimum Broadcasting and Personalized Communication in Hypercubes. *IEEE Transaction on Computers*, 9(38):1249–1268, 1989.

[JM89] P. Jones and A. Murta. Practical Experience of Run-time Link Reconfiguration in a Multi-transputer Machine. In *Concurrency: Practice and Experience*, volume 1. John Wiley and sons, Ltd, December 1989.

[JMM92] D.B. Johnson, F. Makedon, and P. Metaxas, editors. *Proc. of the DAGS/PC symp.* 1992.

[KA91] A.E. Knowles and N.N. Avramov. A Message Passing System for a Network of Transputers. In K. Boyanov, editor, *Parallel and Distributed Processing*, pages 27–39. Elsevier Science Publishers B.V. (North-Holland), 1991.

[Sei89] S.R. Seidel. Circuit-Switched vs. Store and Forward Solutions to Symmetric Communication Problems. In *HCCA 4*, 1989.

[SS89] Y. Saad and M.H. Schultz. Data Communication in Parallel Architectures. *Journal of Parallel and Distributed Computing*, 6:115–135, 1989.

[SW87] Q.F. Stout and B. Wagar. Intensive Hypercube Communications. Technical report, University of Michigan - Computing Research Laboratory, 1987.

[TCS88] X. Thibault, D. Comte, and P. Siron. A Reconfigurable Optical Interconnection Network for Highly Parallel Architecture. In *Frontiers of Massively Parallel Computation*. IEEE Computer Society Press, 1988.

[TM90] T. Theoharis and J.J. Modi. Implementation of Matrix Multiplication on the T-RACK. *Parallel Computing*, 14:229–233, 1990.

[Wai90] P. Waille. Introduction à l'Architecture des Machines Supernode. Technical Report RT-56, LGI-IMAG, Februar 1990.