Scheduling communications on a multi-cluster network

Alfredo Goldman*
Department of Computer Science
University of São Paulo
Rua do Matão, 1010 CEP 05508-900
São Paulo — Brazil
e-mail: gold@ime.usp.br

Abstract

Nowadays, with the introduction of clusters of computers the notion of interprocessor communication is of growing interest. Inside a dedicated cluster the communication times were generally modeled in the same way, independently of which processors communicate. In a network where the links among the computers are heterogeneous this might not be true anymore. Computers that have faster links, or are closer with each other should be able to exchange messages faster. These differences on communication times should be considered, not only for attributing tasks to the processors but also in global synchronization/communication. The goal of this paper is to study the global communication in a network of dedicated clusters. We propose new communication algorithms based on known algorithms and we present simulations on their expected performances.

keywords: Scheduling of communication, BSP-like synchronization, global communication.

1 Introduction

Several parallel computing applications can be viewed as a succession of steps composed of computation followed by communication among all, or subsets, of the processing nodes. One of the main parallel computing models today, BSP [10], is based on this principle.

Among the applications, we can cite, for illustration, parallel domain decomposition and numerical resolution of partial differential equations. In these applications, the adjacent nodes (in the problem context) exchange information after computation with local data, before the next computation phase.

The first works which addressed the communication problem were either too specific, or not realistic. For example, the works [1] and [9] were based on specific interconnection networks. Other papers dealt with messages of unitary, or equal size [7, 8]. Two works assumed more general models, with messages of different sizes and totally connected networks. In [3], Coffman et al studied the problem of exchanging messages of different sizes, but they did not allow preemption on messages sending. In [2], Choi and Hakimi studied the same problem, allowing transmissions of messages on several, not necessary consecutive, sends. But they assumed a synchronous model. Finally, in [5] the authors studied a more realistic model with the exchange of messages of different sizes in an asynchronous network where preemption was allowed.

In the three last works, the communication times were modeled by the linear cost model. This fits well for most of interprocessors communication. In this model, to send a message, there is a startup (α) and the transmission time depends inversely on the bandwidth (γ). The parameters α and γ were the same for all processors pairs.

In this work we intend to introduce another parameter in the model, the main idea being that communication does not have to behave similarly for all processor pairs. The main motivation of this work is the increasingly more popular use of computers clusters, as one easy way to obtain larger clusters is to interconnect already existing clusters or parallel computers [4]. Generally, the communication time inside a computer cluster does not depend on the involved processors, and if it does depend the influence is not very important. However, if we connect computer clusters among them, the communication of computers from different clusters tends to be slower. We intend to provide algorithms for this case.

Among the main applications of this work are: efficient support to redistribute, verify and synchronize distributed data on heterogenous network, support for executing several concrete problems for which parallel or distributed algorithms alternate periods of computing with periods of data exchange, and support for BSP-like [10] environments or

*Supported by Fapesp, proc. 98/06138-2.

applications.

1.1 Paper organization

This paper is organized as follows, in the next section we detail the problem and the adopted model. In Section 3 we present the already known algorithms for a uniform environment. The algorithms for dedicated multi-clusters are presented in Section 4. Simulations are presented in Section 5. We close this paper with ideas for a future work.

2 Problem and Model

2.1 Problem

In order to model a multi-cluster network (MCN), we will introduce a simplified model. We suppose that there are two connection levels. The clusters are connected among them to obtain the MCN. There will be two different kinds of communications, the local ones and the distant ones. The local communications will model the communications inside a same cluster, and the distant communications the exchange of data between different clusters. See an example of the assumed MCN model in Figure 1.

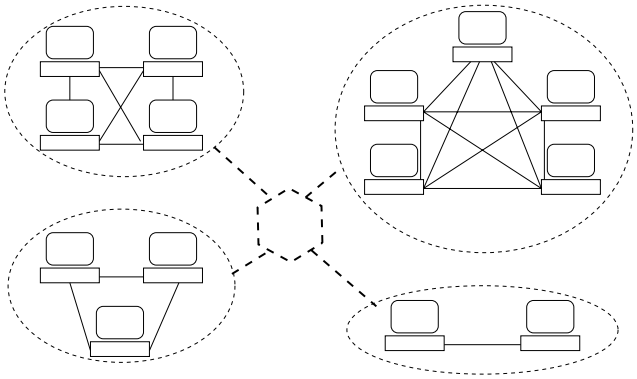

Figure 1. An example of a MCN composed by four clusters, with 2, 3, 4 and 5 computers.

Given a MCN composed by CL clusters, each cluster L_i is composed by $n_i, 0 \leq i < CL$ computers, $L_i = \{p_0^i, \ldots, p_{n_i}^i\}$. We define the set of processors by $\{p_0, \ldots, p_{n-1}\}$ where $n = \sum_{i=0}^{CL-1} n_i$, and p_j^i corresponds to $p_k, k = \sum_{l=0}^{i-1} n_i + j$. Computer p_i has a message of size $m_{i,j}$ to send to computer p_j (if there is no message from p_i to p_j, then $m_{i,j} = 0$).

If several messages are to be sent from p_i to p_j, then $m_{i,j}$ corresponds to the sum of all these messages. We assume that all the messages are known before the algorithms starting time. We represent the problem by a communication matrix $M = (m_{i,j})$. We assume that there is no interprocessor communication, that is $m_{i,i} = 0, 0 \leq i < n$.

An algorithm that solves the message exchange problem guarantees the transmission of all messages to their destination. The total time for such algorithms is the difference between the starting time and the time at which all processors have received all their addressed messages. The goal is to minimize the total time.

2.2 Model

We assume that each computer in the system has a limited capacity of communication, that is, each machine can transmit at most one message and receive at most one message simultaneously. This limitation is well known in the literature as one-port full-duplex. There are computers where the transmission/reception capacity is not limited to one, but the one-port full-duplex hypothesis is interesting because it can also be useful to limit the congestion. With this hypothesis, the number of simultaneous messages in the network is limited to the number of processors.

The communication times will be approximated by the linear model where a transmission of a message of size L takes time $\beta + L\gamma$, where β is the start-up time and γ is the inverse of the bandwidth. We will have different start-ups and bandwidths for the communications. We denote β_l and β_w the start-ups time for transmissions inside the same cluster and between clusters, respectively. We use the same notation γ_l and γ_w for the inverse of the bandwidths. As doing the communications inside a cluster is more efficient than doing inter-cluster communication, we can assume that $\beta_l \leq \beta_w$ and $\gamma_l \leq \gamma_w$.

We also introduce a slowdown factor for communication among clusters. Generally, there exist exclusive communication channels among the processors in a cluster, this is not true for the communication among clusters. If two processors from the same cluster send messages to another cluster, they will share the same communication channel between these two clusters. So the bandwidth of this channel will be smaller than $1/\gamma_w$. If m messages are exchanged simultaneously between the two clusters, the slowdown factor S can be up to m. We do not consider any slowdown for the start-up times.

All the algorithms in this paper describe the communication schedule by a sequence of communication phases. A communication phase is a set of communications such that each processor sends/receives at most one message. When all the messages size are the same, the involved processors send/receive messages during the whole communication phase. Otherwise, there can exist processors which are not involved in communication all the time. The differences

on the message sizes can be more explicit depending on the existence of global synchronizations.

In a synchronous mode, all processors start each communication phase at the same time. So, the next communication phase can start only after the reception of the last message from the previous phase. In an asynchronous model, transmissions from different processors do not have to start at the same time. However, all processors start the algorithm simultaneously.

Below we give examples of schedules for a message exchange problem. We provide one on each model, synchronous and asynchronous, for the following communication matrix:

$$M = \begin{bmatrix} 0 & 9 & 2 & 0 \\ 4 & 0 & 3 & 6 \\ 7 & 5 & 0 & 0 \\ 1 & 0 & 8 & 0 \end{bmatrix}.$$

We represent the sending of messages by a Gantt chart. The x axis represents the time, and the y axis represents the different processors. We could have a similar representation, using the message receptions. In Figure 2 we show the following communication phases:

$$\begin{bmatrix} 0 & 9 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 6 \\ 7 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 \\ 0 & 0 & 8 & 0 \end{bmatrix}.$$

In the schedules of figure 2 we assumed both start-up and bandwidth equals to one. It is interesting to observe that even if processor p_3 is ready to send the second message to processor p_2, from the time instant 2, it cannot send it before because p_2 receives a message from p_0 until the instant 13.

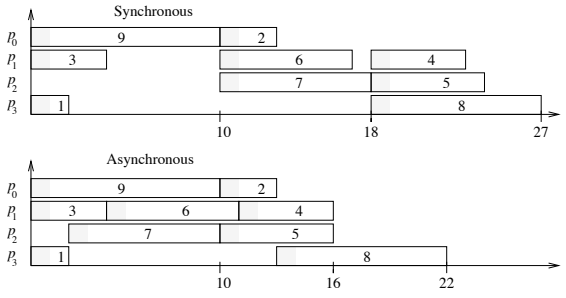

Figure 2. Communication phases in the synchronous and asynchronous models.

An algorithm for exchanging messages can be classified according to the way that the messages transmission is done. There exist two main classifications: message forwarding and message splitting. If each message is sent directly from the source to the destination, without being stored on intermediate processors, there is no message forwarding. When the messages are sent within just one transmission, without cutting the message in smaller pieces, there is no message splitting.

3 Scheduling of communication on a uniform environment

The goal of this section is to give an overview of the known algorithms to schedule communications on a uniform environment. A detailed description of these algorithms can be found in [5].

3.1 FP - Fixed Pattern Algorithm

We present a simple algorithm that does neither message forwarding nor message splitting.

```
for t = 1 to n − 1 do
  do in parallel for all j (0 ≤ j < n)
    p_j sends m_{i,j+t mod n} to p_{j+t mod n}
    p_j receives m_{j,j−t mod n} from p_{j−t mod n}
```

Each processor executes $n-1$ communication phases in this algorithm. In each communication phase t, processor p_i sends its message to processor $p_{i+t \bmod n}$.

3.2 HL - Hypercube-like Algorithm

This algorithm was first proposed for hypercubes. This algorithm does message forwarding and does not split the messages. For ease of description, we first present the algorithm assuming that the number of processors is a power of 2 ($n = 2^l$). Each processor is labeled with a binary string and we use $p_i^{\overline{j}}$ to denote the processor whose label differs from the label of processor p_i only in the j^{th} bit position.

```
for t = 1 to l do
   do in parallel for all i (0 ≤ i < n)
       p_i exchanges messages with p_i^{\overline{t}}
```

An extension of the previous algorithm for an even number of processors can be found in [6].

3.3 Matrix Algorithms

The main idea of the matrix algorithms is to provide the knowledge of the whole communication matrix on each processor, and then to apply an algorithm which provides better communication phases. We expect that the extra cost to provide the global knowledge can be recovered by doing the communication faster. The algorithms can be written as:

```
all-to-all exchange of the message sizes
global strategy determination
strategy execution
```

The last two stages can be merged in order to overlap the strategy determination processing time by the communication on the strategy execution.

3.3.1 MM - Max-Min algorithm

The Max-Min algorithm computes communication phases where the smallest message size is maximized. The algorithm works by steps. In the first step a max-min communication phase is found in the original communication matrix. In the next steps the communication phases are computed on the updated communication matrix, that is, the messages that appear on the previous communication phases are considered as zero size messages.

```
all-to-all exchange of the message sizes
while the comm. matrix M is not done
  find a Max-Min communication phase P
  send the messages on P
  subtract the messages on P from M
```

3.3.2 MS - Max-Sum algorithm

The Max-Sum algorithm is similar to the Max-Min algorithm, on both an optimization criterion is used on the communication phases. Neither MM nor MS does message forwarding or message splitting. On the Max-Sum algorithm the sum of the messages size is maximized on each communication phase.

3.3.3 Unif - Uniform algorithm

The main idea of the uniform algorithm is to obtain communication phases where all the messages have the same size. This algorithm requires a pre-processing phase to obtain a special communication matrix.

Let α be the largest sum among the elements in the same row, or in the same column of the communication matrix. First a communication matrix, Q, where all rows and columns have the same sum value α is computed. The matrix Q is obtained from the original communication matrix by adding dummy messages. Note that in this new matrix, all nodes have to transmit (and also receive) messages for which the sizes sum to α.

We compute a sequence of communication phases on Q such that on each one, the messages have the same size. This can be done finding a communication phase where the smallest message is minimized, and then normalizing it by assuming all the messages with this smallest size. The obtained communications are then subtracted from the communication matrix. Obviously, there is no need to do the transmissions corresponding to the added dummy messages when the algorithm is executed.

```
all-to-all exchange of the message sizes
find a quasi-doubly stochastic matrix Q
while the comm. matrix Q is not done
  find a Max-Min communication phase P
  normalize P by the smallest message
  send the ''real'' messages on P
  subtract the messages on P from Q
```

3.4 Differences on the performance of the algorithms

In Figure 3 we present the schedules generated by the uniform algorithms for the matrix M given at the end of Section 2. The schedule generated by FP was already presented in Figure 2. In the Gantt chart of HL the messages which are forwarded are denoted with an f.

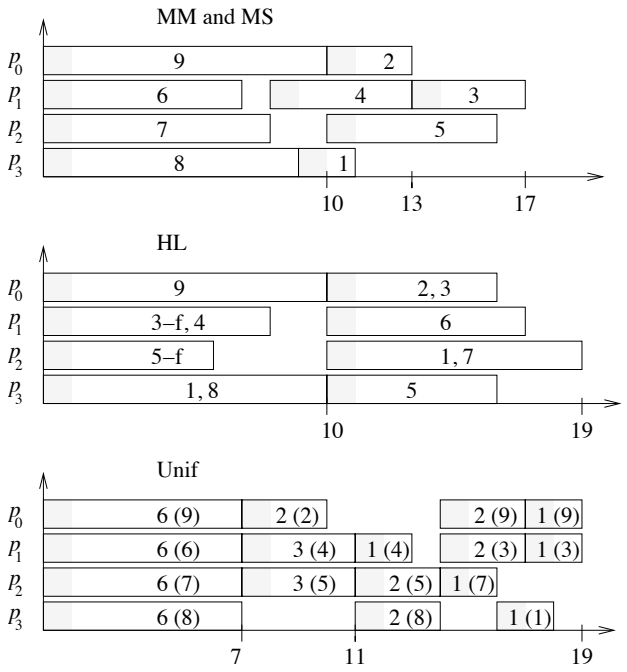

Figure 3. Uniform algorithms schedules.

The choice of the most suitable algorithm first depends on the parallel processing support. In an environment based on the message exchange paradigm, like MPI or PVM, there are primitives like send and receive. On these primitives, there is a parameter that is the message size. So, in this case, the first step before any communication algorithm is to exchange the size of the messages to be exchanged, and the overhead to provide the global knowledge can be ignored. In other kind of environments, like the one with read

and write functions, this cost is already included in the communication primitives. So, in this case, we have to consider the overhead to provide the global knowledge.

In a given environment the choice depends on the parameters of the communication cost model (β and τ), and on the message sizes. HL minimizes the number of start-ups, and Unif minimizes the bandwidth waste. When neither the start-up is too big nor to small, the choice of MM, MS or FP depends on the variance on the messages size. If this variance is small it is not worthwhile to provide the global knowledge (which involves communication) in order to find more efficient communication phases. Otherwise, the total time of MM or MS is usually better that the time of FP.

It is not easy to distinguish between MM and MS strategy performances. Both algorithms were implemented on a parallel computer, and MS performed slightly better than MM in experiments [5]. The differences in performance depend not only on the variance on the message size, but also on the message sizes distribution.

4 Scheduling of Communications on MCN

Obviously, the algorithms of the previous section can be used, without any change for a MCN. Unfortunately, due the differences on the local and distant communications, the algorithms performance will be degraded. In this section we propose two strategies for scheduling MCN communications, the first one is to adapt the previous algorithms to the new environment. The second one proposes an algorithm which perform first a scheduling on each cluster, then perform the scheduling among them.

Among the algoritms of this section, only the last one considers the slowdown factor implicitly.

4.1 Adapting the algorithms

4.1.1 FP and HL

As the fixed pattern and hypercube like algorithms do not use the information on the message sizes, they cannot be adapted to the new environment. As the notion of distant communications introduces a new heterogeneity, one could expect that the performance of these algorithms will be influenced by the differences between local and distant communication.

4.1.2 MM and MS

On the other side, the other algorithms can be adapted. To do that, we have to introduce additional costs for the messages to be transmitted between clusters. To implement this cost, we have to consider the differences on the communication parameters. Instead of using the communication matrix we will use the expected time matrix. If a message of size L has to be send inside a cluster, the communication cost will be represented by $\beta_l + L\gamma_l$ on the time matrix. Otherwise, if this message involves a distant communication it should be represented by $\beta_w + L\gamma_w$.

The communication phase is found using the time matrix, but only the information on the communication pattern (the pairs of processors involved on transmission-reception) is used. The algorithm is given below:

```
build from M the expected time matrix T
while the comm. matrix M is not done
  find a communication phase C_T on T
  use the comm. pattern C_T to construct C_M
  send the messages on C_M
  subtract from M the messages on C_M
```

4.1.3 Unif

The substitution of the communication matrix by a time matrix can be applied directly to MM and MS, but in order to use Unif we have to search a more complicated strategy. To be able to use the same idea we should know the number of times that each message will be split before the beginning of the algorithm.

We propose a different way to solve this problem. The main idea is to reconstruct the time matrix after each communication phase. In each communication phase, each transmission inside a cluster uses a local communication, the transmissions between clusters use a distant communication. The differences on these communications have to be considered in a normalization phase. Let t be the smallest time on a communication phase, there are two upper bounds on the messages size, m_l and m_w for local and distant communication, such that $t = \beta_l + m_l\gamma_l = \beta_w + m_w\gamma_w$. So the local messages size is limited to m_l, and the distant messages size limited to m_w.

We use the following algorithm:

```
build from M the expected time matrix T
while the comm. matrix M is not done
  find a communication phase C_T on T
  use C_T comm. pattern  to construct C_M
  normalize C_M
  send the messages on C_M
  subtract from M the messages on C_M
  reconstruct T from C
```

4.2 Two level algorithms

The main idea is to solve the problem level by level, first the messages are exchanged on each cluster, and messages to each other cluster are centralized on one node. Then the nodes that received the messages for other clusters exchange the messages on the MCN. Finally these nodes broadcast the messages to the final destinations.

```
message exchange on each cluster
message exchange among clusters
message distribution on each cluster
```

Any of the algorithms of Section 3 can be used in the first two steps. The best choice depends on the parameters for local and distant communication. A two-level algorithm does message forwarding, it does message splitting depending on the choice of the algorithm. For each phase of the two-level algorithm we choose the algorithm with the best expected performance.

In order to improve the third phase, we assume that in each cluster L_i, composed by the processors $\{p_0^i, \ldots, p^i\}$, the processor p_j^i is the one that performs communication with the cluster L_j. In this way, after the message exchange among clusters, the messages to be distributed on each cluster will be located on $\min\{|L_i|, CL\}$ nodes.

5 Simulation

We divide the simulation section into several subsections. First we show for a given example the degradation of the performance of the uniform algorithms as the ratio distant communication over local communication increases. Then we analyze the performance of the new algorithms on the same example. Finally we compare the performance of the algorithms in several situations. The influence of the slowdown factor will be analyzed only in the fourth subsection.

The initial costs for the exchange of the messages size are not considered to compute the total time of the algorithms. The difference between this initial cost and the cost to provide global knowledge on the messages size, is usually very small, for example this difference is less than $\lceil \log_2 n \rceil \frac{n}{2}(n-2)s\gamma_w$ using HL in a message passing environment, where n is the number of processors, and s is the number of bytes used to store the message size.

5.1 Performance of uniform algorithms on MCNs

We show below the behavior of the uniform algorithms on a cluster, the parameters used for the start-up and bandwidth were obtained from a Myrinet network. Start-up is equal to 800 microseconds and the inverse of the bandwidth is 0.28 microseconds. We assumed messages given from a Gaussian distribution where the average message size is 100000 and the variance is 100000. The first simulation was done with 12 nodes divided into clusters of size 3.

The total times in milliseconds for the uniform algorithms are given on the next table.

FP	HL	MM	MS	Unif
569	859	610	542	508

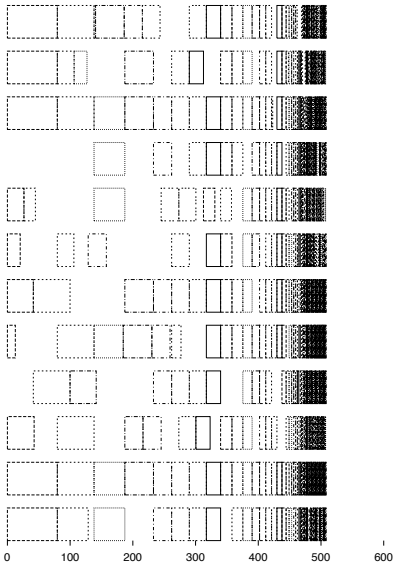

Figure 4. Communication phases of Unif on an uniform network.

For this given example as the start-up time is small, the best performances were obtained with Unif. In the following figures each line corresponds to the transmissions of a processor, the rectangles correspond to the messages sending, the different phases can be viewed on the rectangle borders.

In Figure 4 the behavior of Unif is depicted. It is easy to see that when there is communication, the communication phases have messages of similar sizes.

Now we assume with the same communication matrix as in the previous example that the distant communication is 10 times slower than the local communication, that is $\beta_w = 10\beta_l$ and $\gamma_w = 10\gamma_l$. The total times in milliseconds for the uniform algorithms are:

FP	HL	MM	MS	Unif
5043	8591	5233	4915	4711

We can easily notice that the total time of the algorithms was up to 10 times larger, that is having faster local communication just provide a small improvement (up to 15%). The communication phases of FP and Unif are shown in the Figures 5 and 6. We can notice that the communication phases of Unif do not have messages of similar size.

In the next table we present the total time for the uniform algorithms when the distant communication is 100 times and 1000 times slower than the local communication. The times are given in seconds.

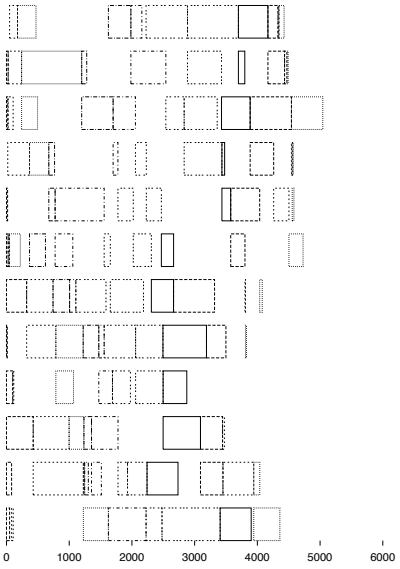

Figure 5. Communication phases of FP on a MCN where the local communication is ten times faster than the distant one.

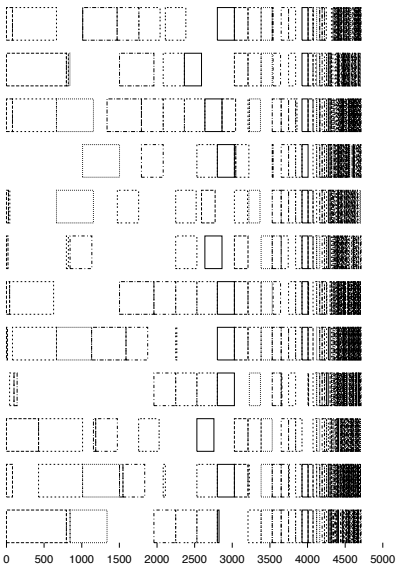

Figure 6. Communication phases of Unif on a MCN where the local communication is ten times faster than the distant one.

FP	HL	MM	MS	Unif
50,43	85,91	51,46	49,12	47,11
504,3	859,1	513,7	491,3	471,1

From these results we can observe that when the distant communications become more time costly, having fast local communications did not improve the algorithms performance.

5.2 Performance of uniform algorithms on MCNs

In the next table we can see the performances of the adapted algorithms on the communication matrix of previous subsection. Again, as the start-up cost is relatively small the Unif algorithm, which splits messages, has the best performances. When the distant communication cost up to 10 times the local communication, the times are given in milliseconds. In the other two cases the cost is given in seconds. For these simulations the slowdown factor was not considered.

γ_w/γ_l	MM	MS	Unif	Two level
1 - (ms)	610	542	508	1767
10 - (ms)	4657	4277	4027	9875
100 - (s)	47,1	41,9	39,9	92,6
1000 -(s)	471,5	418,2	398,3	919,7

The better performance of the adapted algorithms is due to more uniform communication phases. These algorithms take advantage of the faster local communication by doing phases where there are only local communications, only distant communications, or large local messages and small distant messages. In Figure 7 we can see that the phases have messages of similar sizes, which is not the case of Figure 6.

The bad performance of the two-level algorithm can be explained by the small influence of the start-up, which is small, which favors the algorithms that split messages instead of the message forwarding algorithms. When the influence of the start-up is more important, the two level algorithm performs better.

The example below uses the communication parameters of the IBM-SP2 for the local communication: start-up equals to 1.3 milliseconds and the inverse of the bandwidth 0.035 microseconds. For the distant communication we just assume a larger start-up time, which is 10 times larger. The messages are obtained from a Gaussian distribution with average size 10000 and variance 10000. We use 20 processors divided into five clusters of size four. The algorithms performance in milliseconds is:

	FP	HL	MM	MS	Unif	2-level
uniform	259	88	271	258	1270	
adapted			258	224	574	113

Even in this example, the performance of the two level algorithm is not as good as HL, which performs very well for big start-up times. The reason for this can be seen

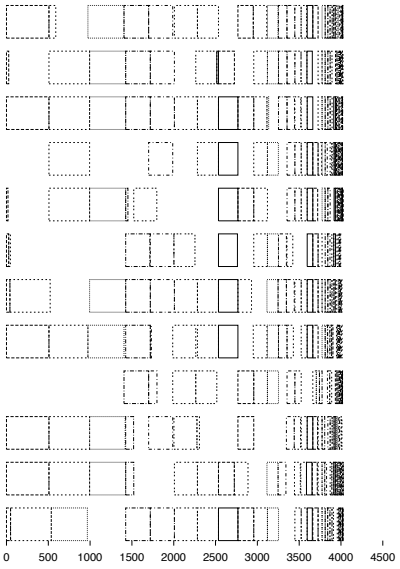

Figure 7. Communication phases of adapted Unif on a MCN where the local communication is ten times faster than the distant one.

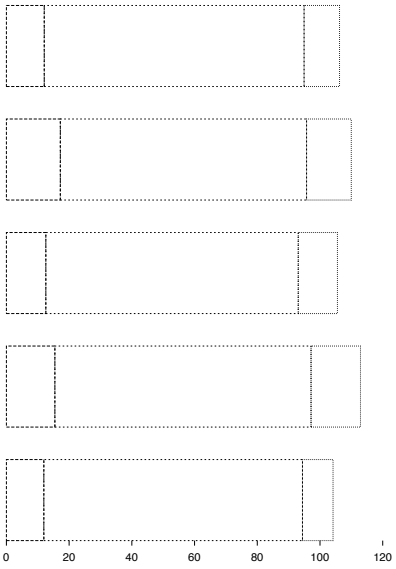

Figure 8. Macro communication phases of two-level algorithm.

comparing Figures 8 and 9. In Figure 8 we can see in the first communication phase the exchange of messages in each cluster, in the second phase the exchange of messages among clusters. Finally, the third phase distributes the messages inside each cluster.

In the second phase of the two-level algorithm as the exchange is composed of big messages, the message forwarding is costly. In this example, MM was used in the second phase (it performed slightly better than HL).

5.3 Performance of the algorithms

In order to analyze performance of the algorithms, we will simulate several situations: small, medium and big clusters; medium and large number of processors; small, medium and big messages. In order to eliminate particular situations, for each situation we compute the average of 20 experimentations. As we do not intend to compare the algorithms among them, we use fixed parameters for the start-up and bandwidth. For local communication we use the Myrinet parameters (the start-up is 800 microseconds and the inverse of the bandwidth is 0.28 microseconds). We assume distant communications 10 times slower.

5.3.1 Small clusters

In this subsection the simulations are done with clusters composed by three processors. For 12 processors and small messages (average 1000, and variance 1000) we obtained the following results (in milliseconds):

	FP	HL	MM	MS	Unif	2-level
uniform	140	131	138	139	430	
adapted			127	122	540	133

As expected the best algorithms for small messages are those which minimize the number of phases, or use message forwarding. Both versions of Unif have bad performances. In the following table the only difference from the previous example are messages 10 times larger (medium messages).

	FP	HL	MM	MS	Unif	2-level
uniform	615	1025	609	585	943	
adapted			571	542	945	1087

It is not worthwhile anymore to do message forwarding in order to minimize the number of phases. The best performance is obtained with adapted MS. For the next example the 20 instances were obtained with message average and variance equals to 100000 (big messages). The following results are given in seconds.

	FP	HL	MM	MS	Unif	2-level
uniform	5,40	9,96	5,35	5,07	5,34	
adapted			5,02	4,75	4,86	10,57

In this case the algorithms with a good balance on the communication phases have good performances, even if they have several communication phases as the uniform algorithm.

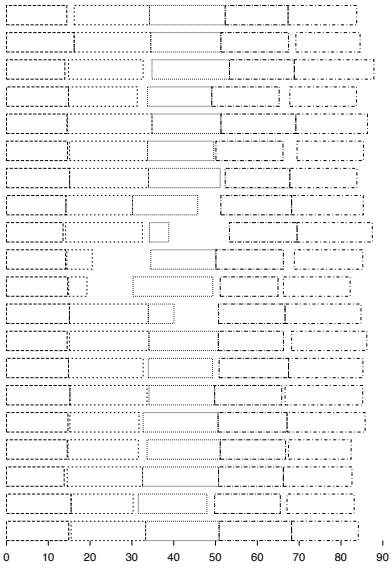

Figure 9. Communication phases of HL.

5.3.2 Medium clusters

In this subsection the simulation is done in clusters of 8 computers. The number of nodes is 64. Here even for small messages the best performances were obtained with adapted MM. The following two tables are in milliseconds.

	FP	HL	MM	MS	Unif	2-level
uniform	831	741	747	739	2176	
adapted			714	678	1309	1588

For medium and big messages the results are (in seconds):

	FP	HL	MM	MS	Unif	2-level
uniform	3,82	6,98	3,25	3,09	4,80	
adapted			3,12	2,91	3,78	15,22
uniform	33,8	694,1	28,4	26,7	26,8	
adapted			27,2	25,2	23,9	150,0

5.3.3 Big clusters

We also simulate a MCN with 4 clusters of 32 nodes. The obtained results for small, medium and big messages are (in seconds):

	FP	HL	MM	MS	Unif	2-level
uniform	1,66	1,61	1,43	1,42	4,30	
adapted			1,24	1,17	1,82	10,17
uniform	7,54	15,6	5,95	5,79	9,45	
adapted			5,30	4,94	5,99	101
uniform	66,5	155,3	51,6	49,6	51,5	
adapted			46,0	42,5	40,1	1004

The best algorithms are those which have better communication phases. When the messages size increase, the start-up time becomes less important, so for small and medium messages the best results are obtained with MM, and for big messages Unif provides the smallest message exchange time.

5.4 Slowdown factor

The slowdown factor can be important when the number of processors in each cluster is big. With the same parameters as in the previous subsection we assumed a maximal slowdown factor. That is, if more than one message share the same link between LANs, the bandwidth is divided by the number of messages sharing the link. The times in seconds are:

adapted	MM	MS	Unif	2-level
small	14,6	13,3	13,7	10,2
medium	139	126	115	101
big	1403	1275	1119	1004

As we can see in the previous results, when the slowdown factor is big the best results are given by two-level algorithms.

6 Conclusion

The algorithms proposed for the exchange of messages on MCNs provide performances up to 20% better than the usual algorithms.

The MS provides good performances even in a heterogeneous environment, so as we could expect adapted MS is the algorithm which provides the best results when the start-up time is important.

For future works it would be interesting to generalize the MCN model to consider more than two kinds of communications, each one whit its own start-up and bandwidth. It would also be interesting to find more realistic slowdown functions, which should reflect the behaviour of actual networks of computers.

References

[1] S.H. Bokhari. Multiphase complete exchange on a circuit switched hypercube. In *Proceedings of the 1991 International Conference on Parallel Processing*, volume I, pages 525–529, 1991.

[2] H. Choi and S.L. Hakimi. Data transfer in networks. *Algorithmica*, 3:223–245, 1988.

[3] E.G. Coffman, M.R. Garey, D.S. Johnson, and A.S. Lapaugh. Scheduling file transfers. *SIAM J. Comput.*, 14(3):744–780, August 1985.

[4] I. Foster and C. Kesselman, Globus: A Metacomputing Infrastructure Toolkit. *Intl J. Supercomputer Applications* , 11(2):115-128, 1997.

[5] A. Goldman, J. Peters, and D. Trystram. Exchange of messages of different sizes. In *IRREGULAR'98*, Lecture Notes in Computer Science 1457, Springer-Verlag, pages 194–205, 1998.

[6] A. Goldman. *Impact des modèles d'exécution pour l'ordonancement en calcul parallèle*. Thesis, Institut National Polythecnique de Grenoble, 1999.

[7] T.F. Gonzales. Multi-message multicasting. In *IRREGULAR'96*, Lecture Notes in Computer Science 1117, Springer-Verlag, pages 217–228, 1996.

[8] S. Ranka, R.V Shankar, and K.A. Alsabti. Many-to-many personalized communication with bounded traffic. In *The Fifth Symposium on the Frontiers of Massively Parallel Computation*, pages 20–27, Feb., 1995.

[9] R. Tahkur and A. Choudhary. All-to-all communication on meshes with wormhole routing. In *IPPS'94*, pages 561–565, 1994.

[10] L.G. Valiant. A bridging model for parallel computation. *Communications of the ACM*, 33(8):103–111, August 1990.