Implementation and evaluation of distributed synchronization on a distributed memory parallel machine

André Couvert, René Pedrono, Michel Raynal

INRIA

UNITÉ DE RECHERCHE
INRIA-RENNES

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P.105
78153 Le Chesnay Cedex
France
Tél.:(1) 39 63 55 11

Rapports de Recherche

N° 1280

Programme 3
Réseaux et Systèmes Répartis

IMPLEMENTATION AND EVALUATION OF DISTRIBUTED SYNCHRONIZATION ON A DISTRIBUTED MEMORY PARALLEL MACHINE

André COUVERT
René PEDRONO
Michel RAYNAL

Août 1990

RR_1280

IRISA INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTEMES ALEATOIRES

Campus Universitaire de Beaulieu
35042 - RENNES CEDEX
FRANCE
Téléphone : 99.36.20.00

Implementation and Evaluation of Distributed synchronization on a distributed memory parallel machine

André COUVERT, René PEDRONO, Michel RAYNAL
IRISA
Campus de Beaulieu
F-35042 Rennes Cédex
FRANCE
raynal@irisa.fr

July 23, 1990

Publication Interne n° 544 - Juillet 1990 - 14 Pages

Abstract

Advent of distributed memory parallel machines make possible to study and analyze distributed algorithms in a real context. In this paper we are interested in a paradigm of distributed computing : the implementation of (binary and multiway) rendez-vous. This problem actually includes two sub-problems encountered in several synchronization problems : how to realize a coordination (of the processes involved in the rendez-vous) and how to ensure some exclusion (between conflicting rendez-vous sharing some processes). Several algorithms implementing rendez-vous are presented. Implementations of these protocols on an hypercube are analyzed and compared according to a certain number of parameters ; an efficiency ratio is introduced in order to make these comparisons easier. In addition to the results exhibited, this paper suggests a way to conduct such experiments.

Implémentation et Evaluation d'un schéma de synchronisation répartie sur une machine à mémoire distribuée

Résumé.

L'apparition des machines parallèles à mémoires distribuées rend possible l'étude et l'analyse en vraie grandeur des algorithmes répartis. On s'intéresse dans cet article à un paradigme de cette classe d'algorithmes : la mise en oeuvre des rendez-vous (binaire et multiple). Ce problème présente en effet les 2 problèmes sous-jacents à de nombreux problèmes de synchronisation : réaliser une coordination (des processus en rendez-vous) et une exclusion (entre rendez-vous conflictuels). Divers algorithmes sont présentés; leurs implémentations sur un hypercube sont analysées et comparées en fonction d'un certain nombre de paramètres ; une mesure d'efficacité est introduite afin de faciliter cette comparaison. Outre les résultats exhibés, cet article propose ainsi une démarche pour de telles expérimentations.

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE
(U.R.A. 227)
UNIVERSITE DE RENNES I I.N.S.A. DE RENNES

INSTITUT NATIONAL DE RECHERCHE
EN INFORMATIQUE ET EN AUTOMATIQUE
(UNITE DE RECHERCHE DE RENNES)

Implementation and Evaluation of Distributed synchronization on a distributed memory parallel machine

André COUVERT, René PEDRONO, Michel RAYNAL
IRISA
Campus de Beaulieu
F-35042 Rennes Cédex
FRANCE
raynal@irisa.fr

July 23, 1990

Abstract

Advent of distributed memory parallel machines make possible to study and analyze distributed algorithms in a real context. In this paper we are interested in a paradigm of distributed computing : the implementation of (binary and multiway) rendez-vous. This problem actually includes two sub-problems encountered in several synchronization problems : how to realize a coordination (of the processes involved in the rendez-vous) and how to ensure some exclusion (between conflicting rendez-vous sharing some processes). Several algorithms implementing rendez-vous are presented. Implementations of these protocols on an hypercube are analyzed and compared according to a certain number of parameters : an efficiency ratio is introduced in order to make these comparisons easier. In addition to the results exhibited, this paper suggests a way to conduct such experiments.

1 Introduction

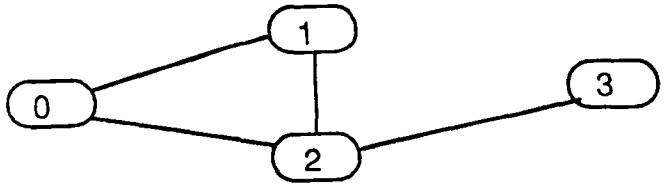
Advent of distributed memory parallel machines make possible to study and analyze distributed algorithms in a real context. such studies are interesting from two points of view. On the one hand they allow to compare distinct algorithms implementing the same function (or service) in terms of efficiency and computation time ; on the other hand they allow to gain knowledge about the mastering of these machines. This paper is concerned with these two aspects.
Here we are interested in a class of distributed synchronization algorithms : protocols implementing (so called binary or multiway) rendez-vous [HOA 78, CM 88, BUR 88, BAG 89b, C 87]. The problem is to allow a certain number of processes to synchronize in some points of their respective computations in such a way (C1) that all the processes involved in a rendez-vous are simultaneously at their meeting point and (C2) that a process is involved in at most one rendez-vous at any time. A rendez-vous can involve only 2 processes as in CSP[HOA 78] or OCCAM[BUR 88] (such rendez-vous are called binary) or any number k of processes

(multiway or k-ary rendez-vous also called committee coordination problem) [CM 88,BAR 89b]. The interest of the rendez-vous lies in its usefulness to solve practical problems [HOA 78, C 87, CM 88] as well as in the fact it grasps two important synchronization problems encountered in many situations : a rendez-vous implies a *coordination* between 2 (or k) processes (synchronism or mutual coincidence of the processes willing to participate in a same rendez-vous : C1) and a process can be involved at the same time in only one rendez-vous chosen in the set of rendez-vous it has announced to be non-deterministically interested in (*mutual exclusion* between conflicting rendez-vous : C2). As such the rendez-vous is a paradigm of distributed systems control problems.
Two protocols (or distributed algorithms) implementing rendez-vous are studied. The first one is devoted to binary rendez-vous [BAG 89a] ; the second one to multiway rendez-vous [BAG 89b]. These algorithms are briefly described in the second part. The third part first gives the experimentation context : the underlying machine is a 64 processors Intel hypercube on which an implementation of the ISO programming language Estelle [ISO 86] has been developped [JJ 89] ; then the implementation choices concerning the two algorithms are presented. The parameters of the analysis and an efficiency ratio to allow comparisons are presented and explained in the fourth part. Part 5 presents and analyses the results of the experiments and the conclusion draws lessons from this experimentation.

2 Studied Algorithms

2.1 Binary rendez-vous

Each process of the distributed program (composed of n processes) is endowed with a manager. The set of managers has to establish rendez-vous described by rule (C1) without violating rule (C2). As an example let us consider the following *rendez-vous graph* between the processes P_0, P_1, P_2, P_3.

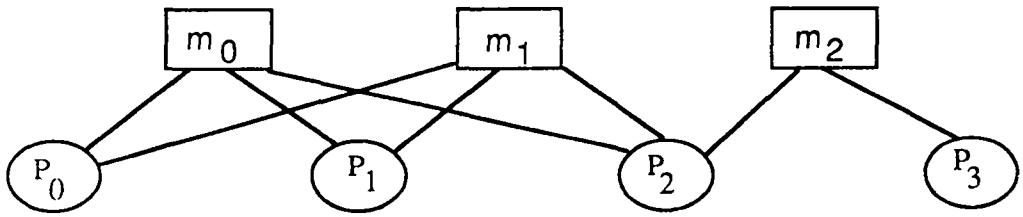


A possible rendez-vous between processes P_i and P_j is represented by the undirected edge *(i,j)* ; such a graph describes the possibilities of rendez-vous at a given time. In the example the process P_0 (resp. P_2) is willing to establish a rendez-vous non-deterministically either with process P_1 or with process P_2 (resp. P_0 or P_1 or P_3).
The algorithm [BAG 89a] associates a unique token to every possible rendez-vous; the token associated to *(i,j)* is noted *token(i,j)* and is initially owned by one of P_i or P_j. The manager associated to P_i tries to establish the rendez-vous *(i,j)* only if it has the corresponding token *token(i,j)* ; then it sends it to P_j's manager. If this one wants to establish the *(i,j)* rendez-vous it sends back the token to P_i's manager and they commit rendez-vous *(i,j)*. In the other case P_j's manager answer it with *no* and keeps the token (it will have the initiative to request the next *(i,j)* rendez-vous) ; on receiving *no*, P_i's manager will try to establish an

other rendez-vous *(i,x)* it is interested in if it owns the associated token *token(i,x)*. Additionnal rules avoid deadlock situations (in wich rendez-vous are possible but none is established). The interested reader will report to [BAG 89a] for more details.
The association of a unique token to each potential rendez-vous is the basis of this algorithm : the rules define how these tokens are managed. This algorithm is called *BIN* in the following. Other algorithms implementing binary rendez-vous are described in [R 88] (chapter 4).

2.2 Multiway (or *k*-ary) rendez-vous

In this case the algorithm [BAG 89b] considers P managers ($1 \leq P \leq n$) which have to cooperate in order to establish rendez-vous noted *(i,j,k,l,...)* (the size of this tuple defines the number of processes involved in the corresponding rendez-vous). These managers m_i are placed on an unidirectionnal ring ; and a valued token moves round on this ring. When a process P_i becomes passive waiting for some rendez-vous it sends the concerned managers the rendez-vous sets *(i,j1,j2,...)*, *(i,l1,l2,...)* it is interested in. When a manager owns the token it determines whether rendez-vous can be committed according to its own values and the values carried by the token ; if it is the case, it informs the involved processes. This algorithm will be called *GEN(k)* in the following ($k \geq 2$ being the biggest size of potential rendez-vous).
Uniqueness of the token ensures mutual exclusion between conflicting rendez-vous [R 86], and managers and token informations allow to enable and to commit some of the rendez-vous.
If we consider this algorithm with k=2, the preceding rendez-vous graph, and 3 managers, we can obtain the following structure (among a set of possible structures) in which an edge represents a link allowing a process and a manager to communicate ; the dotted line represents the ring.



These 3 managers can, for example, manage the following rendez-vous :

m_0 manages *(0,1)* and *(0,2)*.
m_1 manages *(0,2)* and *(1,2)*.
m_2 manages *(2,3)*

Each rendez-vous is managed by at least one manager. In the preceding example other rendez-vous management assignments are of course possible. As one can see the number P of managers is not related to the number n of processes. It is possible to have only one manager (and no token in this case) ; we get a centralized version of the algorithm. At the other extreme it is possible to have n managers each

managing all the rendez-vous : in this case we obtain a very distributed version of the algorithm (distribution by duplication). All the intermediate implementation choices are possible. The interested reader will report to [BAG 89b] for more details.

3 Experiment context

3.1 Environment

The machine on which the experiments have been conducted is an Intel IPSC2 of diameter 6 (64 processors) [Intel 87]. The language in which the different algorithms have been programmed is Estelle [ISO 86, BUD 87], a language normalized by ISO, originally devoted to the specification and implementation of protocols. We have chosen this language as it is a high-level language, well suited to distributed system programming and an Estelle-to-C compiler for the IPSC2 [JJ 89] has been developped. The main feature of the language is the two-level programming : first the implementation of processes and managers by automata triggered by the arrivals of messages received through ports and second the separate specification of connections between input and output ports of these automata.

3.2 Studied implementation

As we aimed at studying efficiency of protocols implementing rendez-vous we consider that the application processes do not have proper activity but for rendez-vous requests. Processes are put on distinct processors of the hypercube.
The implementation of *BIN* corresponds to the description given in §2.1. [BAG 89a]. As we have noticed several implementation choices have to be done for the algorithm *GEN*. Two of then are particularly interesting as they are extreme according to the number of managers ; and they have been implemented :

- a unique manager (put on the same site as P_0). It manages all the rendez-vous. This algorithm is noted : *GEN(k)CENT*.
- n managers m_i ; m_i is put on the same site as P_i and manages all the rendez-vous P_i is involved in. This algorithm is noted *GEN(k)DIST*. (In this case the management of a size k rendez-vous is duplicated on k sites ; the set of managers being connected by a ring with an associated token).

3.3 Rendez-vous graphs

3.3.1 Binary rendez-vous

Processes are designed by $0, 1, \ldots, n\text{-}1$. We have studied potential rendez-vous graphs presenting a regular structure, implemented on cubes of diameter d. Indeed irregular structures disallow simple interpretations and explanations of results. Studied rendez-vous graphs are defined by the following sets R_i associated to each process P_i.

$R_i = \{processes\ among\ which\ P_i\ wants\ to\ establish$
$non - deterministically\ a\ binary\ rendez - vous\}$

- ring R :

$$R_i = \{i-1, i+1\} \quad (+;- : modulo\ n)$$

The corresponding algorithms will be named *BIN-R-n*, *GEN(2)CENT-R-n* and *GEN(2)DIST-R-n* (where n stands for the number of application processes).

- fully connected FC :

$$R_i = \{0, 1, 2, \ldots, n-1\} - \{i\}$$

These algorithms are named *BIN-FC-n*, *GEN(2)CENT-FC-n* and *GEN(2)DIST-FC-n* in the following.

- hypercube H :

$$R_i = \{i \textbf{ xor } 2^j : 0 \leq j \leq d-1\}$$

with the processes identities expressed in the binary notation.

- binary tree T : (**div** is the integer division operator)

$$\begin{array}{l} R_0 = \{1\} \\ R_i = \{i \textbf{ div } 2, 2i, 2i+1\} 1 \leq i \leq (n-1) \textbf{ div } 2 \\ R_i = \{i\ div\ 2\} \qquad 1 + (n-1) \textbf{ div } 2 \leq i \leq n-1 \end{array}$$

3.3.2 Multiway rendez-vous

The structure of potential rendez-vous is now an hypergraph. We have been interested in a regular structure generalizing in some sense the ring of the binary case. The set R_i^k of the set of processes with which P_i can be in k-rendez-vous is chosen of size k (the size of a rendez-vous is k and there are k choices of rendez-vous for P_i). For example for k=4 we get (+,- are *modulo* n) :

$$R_i^{k=4} = \{\{i-3, i-2, i-1\}, \{i-2, i-1, i+1\}, \{i-1, i+1, i+2\}, \{i+1, i+2, i+3\}\}$$

4 Experiment parameters and measures performed

4.1 Binary rendez-vous

4.1.1 Parameters

Each experiment we have realized is defined by the following parameters :

- a potential rendez-vous graph : *R/FC/H/T*.
- a number n of processes : n=*2,8,16,32,64*.
- a rendez-vous duration *RdvD* (once a rendez-vous is committed it lasts this duration). *RdvD* varies between 0 and *28* ms ; this duration is constant and the same for all the processes during an experiment.
- the duration *ExpD* of each experiment has been chosen to *60 s*. This value has been determined from several attemps (it allows to realize several tens of thousands of rendez-vous during each experiment).

4.1.2 Measures

The following measures have been done for the 3 algorithms *BIN-X-n*, *GEN(2)CENT-X-n* and *GEN(2)DIST-X-n* with *X=R/FC/H/T* and *n=8 to 64* :

- the average number of rendez-vous a process has committed : *nbrdv*
- the average duration spent by process within rendez-vous : *nbrdv*RdvD*

These values allow to compute the efficiency of an algorithm in some context (defined by *-X-n*) :

$$\text{efficiency ratio} \quad er = \frac{nbrdv * RdvD}{ExpD}$$

We have $0 \leq er \leq 1$. Moreover the value *1 - er* represents the average ratio of the time spent to establish rendez-vous, i.e. to realize control. The closer of 1 is its *er* value, the better is an algorithm.

4.2 Multiway rendez-vous

In addition to the preceding parameters an experiment requires a value of *k* (size of rendez-vous). Values of *k=2,4,8,16,32* with *n=32* have been experimented on the rendez-vous hypergraph "generalized ring" displayed in the §3.2.2.

5 results and analyzes

5.1 Binary rendez-vous

In addition to the evolution of the efficiency ratio of a given algorithm for a rendez-vous graph we have been interested in comparing *BIN*, *GEN(2)CENT* and *GEN(2)DIST* in order to know if the specialized algorithm *BIN* has some advantage over the general ones with *k=2*.

5.1.1 Algorithms *BIN* and *GEN(2)CENT* :

Figure 1 represents for these two algorithms evolutions of efficiency ratios according to rendez-vous durations *RdvD* for a ring rendez-vous graph of 8 and 64 processes. Figure 2 shows this evolution when the rendez-vous graph is fully connected.
These results show :

- the efficiency ratio increases with the rendez-vous duration.
- for both algorithms the efficiency decreases with the number of processes (i.e. with the number of potential rendez-vous at a given time : *n* for a ring, *n(n-1)/2* for a fully connected network).
- the gap between *BIN* and *GEN(2)CENT* increases with the number of processes (this is due in part to the uniqueness of the manager).

The results obtained with hypercube and binary tree rendez-vous graphs are similar and strengthen this analysis.

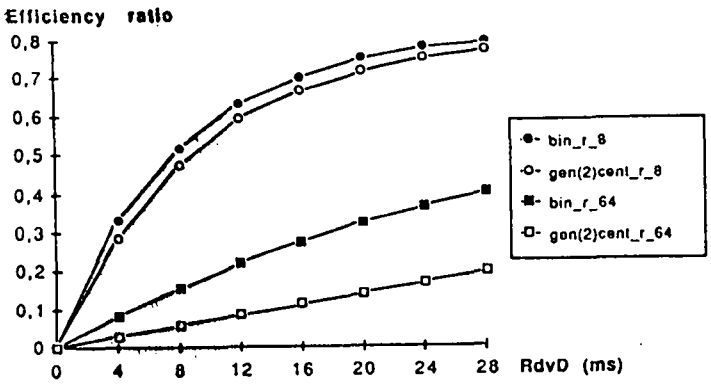


Figure 1 : The Rdv graph is a ring

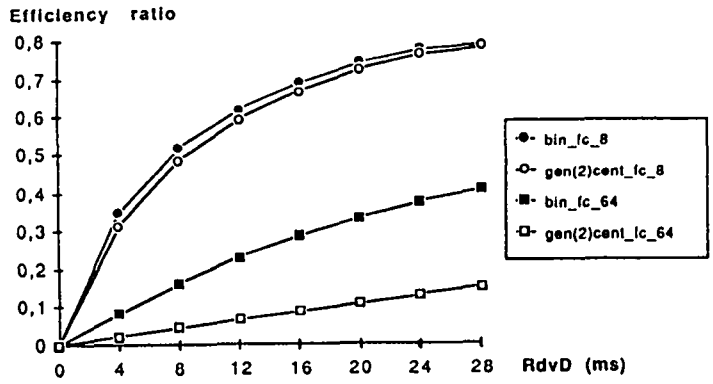


Figure 2 : The Rdv graph is fully connected

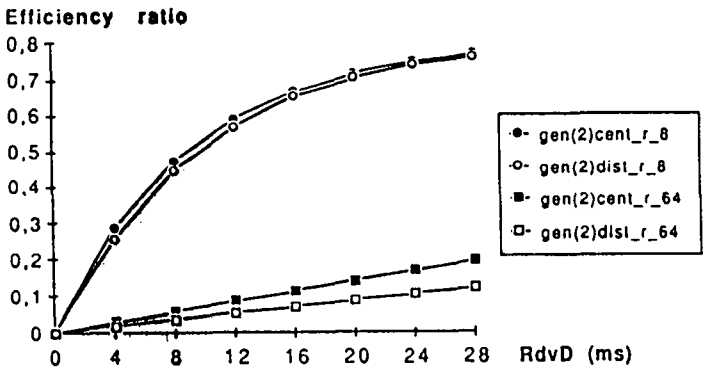


Figure 3 : The Rdv graph is a ring

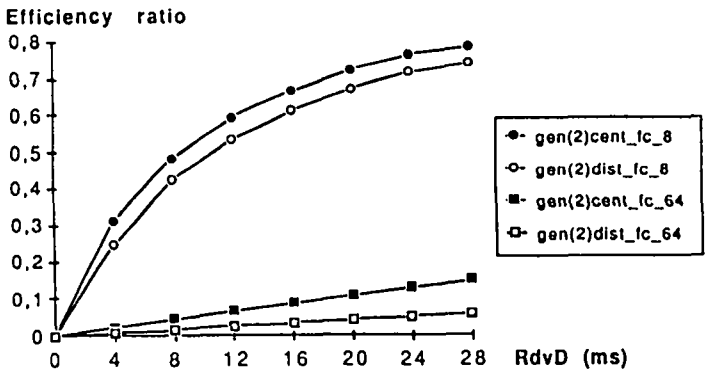


Figure 4 : The Rdv graph is fully connected

5.1.2 Algorithms GEN(2)CENT and GEN(2)DIST

We now compare in the context of the binary rendez-vous the two extreme implementation strategies of the *GEN(k)* algorithm model.
Figures 3 and 4 are analogous to the preceding ones : they compare the efficiency ratios of both algorithms with ring and fully connected rendez-vous graphs of 8 and 64 processors. As previously these results are confirmed by the experiments done on hypercube and binary tree rendez-vous graphs.
The preceding analysis remains valid. The degradation according to n between *GEN(2)CENT* and *GEN(2)DIST* corresponds to the travelling and to the processing time of the token : indeed duplication introduced by the managers can allow to resist some faults but dont improve the parallelism as the token is unique.

5.1.3 Comparison according to the number of processes

Figure 5 gives efficiency ratios of *BIN-R-n*, *GEN(2)CENT-R-n* and *GEN(2)DIST-R-n* (i.e. for a ring rendez-vous graph) according to the total number of processes $n = 2^d$ $(2 \leq d \leq 6)$ and to a rendez-vous duration $RdvD = 20\ ms$.

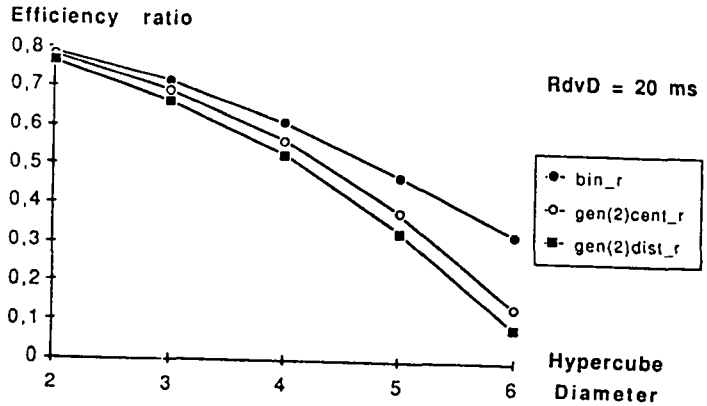


Figure 5 : The Rdv graph is a ring

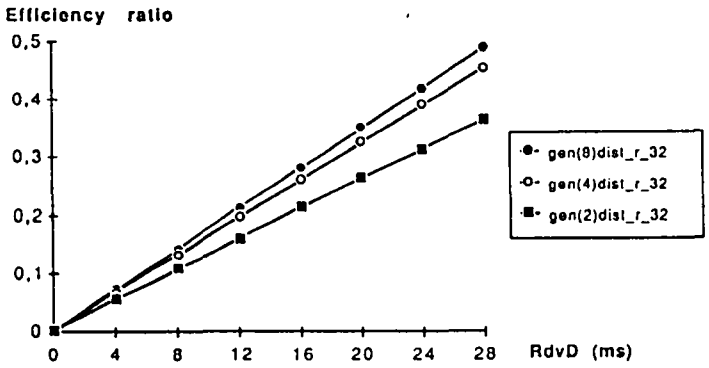


Figure 6 : The Rdv graph is a k-generalized ring

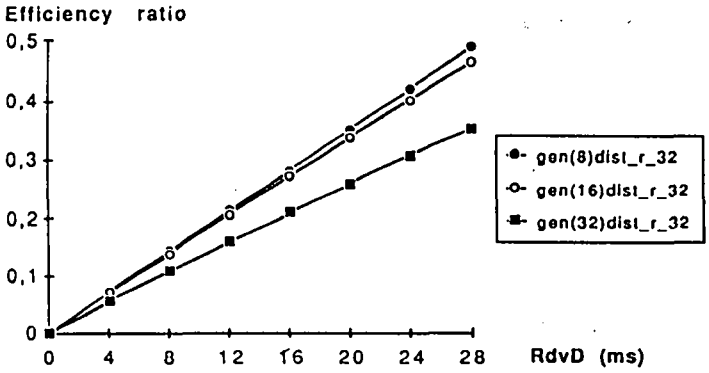


Figure 7 : The Rdv graph is a k-generalized ring

For n=2,4,8 (i.e. $d \leq 3$) the three algorithms have very close efficiency ratios ; for instance for n=8 : *er(BIN-R-8)=0.71*, *er(GEN(2)CENT-R-8)=0.69* and *er(GEN(2)DIST-R-8)=0.66*. On the other hand for n=16,32,64 (i.e. $4 \leq d \leq 6$) the specialized algorithm *BIN* proves to be the most efficient one ; the gap between the two others remains relatively stable ; for n=64, *er(BIN-R-64)=0.32*, *er(GEN(2)CENT-R-64)=0.14* and *er(GEN(2)DIST-R-64)=0.09*.

5.2 Multiway rendez-vous

We are here interested in measuring the efficiency of *GEN(k)DIST* according the size k of multiway rendez-vous. The structure studied for such multiway rendez-vous is the generalized ring described in §3.3.2. (a process asks for a k-rendez-vous in a set of k such potential multiway rendez-vous).

Figures 6 and 7 display efficiency ratios of *GEN(k)DIST-R-32* respectively for k=2,4,8 and k=8,16,32.

On can notice on the one hand that the rendez-vous size k has very little effect on the efficiency (for *RdvD=8 ms* : $0.11 \leq er \leq 0.14$ and for *RdvD=28 ms* : $0.35 \leq er \leq 0.49$). and on the other hand that the best efficiency is obtained for

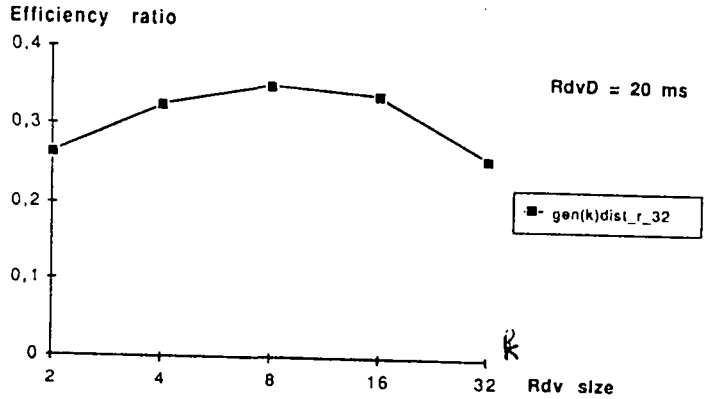


Figure 8 : The Rdv graph is a k-generalized ring

rendez-vous of size $k=8$ (figure 8).

6 Conclusion

The implementation of the (binary or multiway) rendez-vous constitutes a paradigm of distributed control as it presents two of the synchronization major problems : how to realize a *coordination* (of the processes involved in a rendez-vous) and how to realize an *exclusion* (between conflicting rendez-vous). As such it has been chosen to be implemented and evaluated on a distributed memory parallel machine. Several conclusions can be drawn from these experiments. First for binary rendez-vous the specialized algorithm called *BIN* (based on the association of a unique token to each potential rendez-vous) reveals to be always more efficient than the algorithms *GEN(2)*. Then the comparison between *GEN(2)CENT* and *GEN(2)DIST* shows the first is the more efficient one : in other words a centralized control is better in this case. Finally algorithms *GEN(k)DIST* with a distributed control are interesting for rendez-vous of size $k > 2$, as they are relatively stable from an efficiency point of view.
Moreover in addition to the results presented these experiments have allowed a better understanding of the hypercube and have proved the easiness a language such as Estelle provides to realize such an experimentation [A 88].

7 références

References

[A 88] ADAM M., INGELS Ph., JARD Cl., JEZEQUEL J.M., RAYNAL M. *Experimentation on parallel machines is helpful to analyze distributed algorithms.* Workshop Parallel and Distributed algorithms, North Holland, (1988), pp. 243-250

[BAG 89a] BAGRODIA R.L. *Synchronization of asynchronous processes in CSP.* ACM Toplas, vol. 11.4, (Oct. 1989), pp. 585-597

[BAG 89b] BAGRODIA R.L. *Process synchronization : design and performance evaluation of distributed algorithms.* IEEE Trans. on S.E., vol. 15,9, (Sept. 1989), pp. 1053-1065

[BUD 87] BUDKOWSKI S., DEMBINSKI P. *an introduction to Estelle : a specification language for distributed systems.* Computer Networks and ISDN Systems, vol. 14, (1987), pp. 3-23

[BUR 88] BURNS A. *Programming in OCCAM2* Addison-Wesley, (1988), 189 p.

[C 87] CHARLESWORTH A. *The multiway rendez-vous.* ACM Toplas, vol. 9,2, (July 1987), pp. 350-366

[CH 88] CHANDY K.M., MISRA J. *Parallel Program Design : a foundation* Addison-Wesley, (1988), 516 p.

[HOA 78] HOARE C.A.R. *Communicating Sequential Processes.* Comm. ACM, vol. 21,8, (Aug. 1978), pp. 666-670

[INTEL 87] INTEL *Intel IPSC/2 user's guide.* Intel Scientific Computers, Beaverton, (1987)

[ISO 86] ISO *Estelle : a formal description technique based on extended state transition model.* ISO/TC97/SC21/WG16.1/DP 9074,(July 1986)

[JJ 89] JARD Cl., JEZEQUEL J.M. *A multi-processor Estelle to C compiler to experiment distributed algorithms on parallel machines.* Proc. 9th IFIP Int. Workshop on Protocol Specifications and Testing, North-Holland, (1989),

[RAY 88] RAYNAL M. *Distributed Algorithms and Protocols.* Wiley, (1988), 163 p.

[RAY 86] RAYNAL M. *Algorithms for Mutual exclusion.* North Oxford Academic and the MIT Press, (1986), 106 p.

LISTE DES DERNIERES PUBLICATIONS INTERNES

PI 536 **TOWARDS DOCUMENT ENGINEERING**
Vincent QUINT, Marc NANARD, Jacques ANDRE
Mai 1990, 20 Pages.

PI 537 **YALTA : YET ANOTHER LANGUAGE FOR TELEOPERATE APPLICATIONS**
Jean-Christophe PAOLETTI, Lionel MARCE
Juin 1990, 32 Pages.

PI 538 **SYNCHRONOUS DISTRIBUTED ALGORITHMS : A PROOF SYSTEM**
Michel ADAM, Jean-Michel HELARY
Juin 1990, 20 Pages.

PI 539 **CONCEPTION DE DESCRIPTEURS GLOBAUX EN ANALYSE DU MOUVEMENT A PARTIR D'UN CHAMP DENSE DE VECTEURS VITESSES APPARENTES**
Henri NICOLAS, Claude LABIT
Juin 1990, 38 Pages.

PI 540 **DOCUMENT DESCRIPTION LANGUAGE INTERPRESS**
Nenad MAROVAC
Juin 1990, 26 Pages.

PI 541 **UN SIMULATEUR QUALITATIF DE CHAINES BIOLOGIQUES POUR L'AIDE AU DIAGNOSTIC MEDICAL**
Monique LE COZ, Daniel PICART
Juillet 1990, 54 Pages.

PI 542 **A NEW APPROACH TO VISUAL SERVOING IN ROBOTICS**
Bernard ESPIAU, François CHAUMETTE, Patrick RIVES
Juillet 1990, 44 Pages.

PI 543 **SIMPLE DISTRIBUTED SOLUTIONS TO THE READERS-WRITERS PROBLEM**
Michel RAYNAL
Juillet 1990, 10 Pages.

PI 544 **IMPLEMENTATION AND EVALUATION OF DISTRIBUTED SYNCHRONIZATION ON A DISTRIBUTED MEMORY PARALLEL MACHINE**
André COUVERT, René PEDRONO, Michel RAYNAL
Juillet 1990, 14 Pages.

Imprimé en France
par
l'Institut National de Recherche en Informatique et en Automatique

ISSN 0249 - 6399