MINISTERE DES UNIVERSITES ET DE LA RECHERCHE SCIENTIFIQUE

THESE

Présentée au

CENTRE DE DEVELOPPEMENT DES TECHNOLOGIES AVANCEES

Pour l'obtention du grade de

MAGISTER EN CYBERNETIQUE

(Option : Architecture des systèmes)

ETUDE ET EVALUATION D'ALGORITHMES DE CONTROLE DISTRIRUE : ELECTION

Par:

Melle Taboudjemat Nadia

Soutenue le 25 Mai 1992 Devant lo jury Composé de

Mr H. Khelalfa	*Président*
Mr M. Benhamadi	*Examinateur*
Mr M. Benmihoub	*Examinateur*
Mr S. Lezzar	*Examinateur*
Mr N. Badache	*Rapporteur*

A toute ma famille,

et tous mes amis.

Remerciements

-*-*-*-*-*-*-

Je remercie très sincerement:

Mr. H. KHELALFA, chef du laboratoire de recherche et de developpement en informatique(LRDI-CERIST), qui me fait l'honneur de presider le jury de cette these.

Mr. M. BENHAMADI, Directeur du CERIST, pour avoir mis a ma disposition les moyens necessaires a la bonne marche de mon travail, pour la confiance qu'il m'a témoignée, et sa participation au jury.

Mr. M. BENMIHOUB, charge de recherche au laboratoire intelligence artificielle au CDTA, Mr. S. LEZZAR, chef du laboratoire de genie logiciel au CDTA, qui ont accepté de participer au jury de cette these.

Je tiens à exprimer ma reconnaissance à Mr. N. BADACHE, charge de cours à l'USTHB qui m'a été d'une aide precieuse en dirigeant mon travail.

Je remercie l'ensemble du personnel du CERIST pour leur aide et sympathie.

Que tous ceux qui m'ont soutenueet encouragée trouvent ici l'assurance de mes remerciements sincères et amicaux.

RESUME

La nature complexe des algorithmes distribues engendre, dans certains cas, une discordance entre les resultats theoriques et les resultats pratiques. Parfois des mises en oeuvres(même simulées) permettent de deceler des comportements inattendus, voire incorrects.

Cette these est consacree a 1' etude des algorithmes d'election. Le problème est, en effet, fondamental dans le calcul distribue. Les applications qui utilisent ce mecanisme sont nombreuses et variees.

Le but de cette these a été de faire un inventaire de ces algorithmes, en les repartissant par classes en fonction de la topologie du réseau. Leur specification en Estelle, leur evaluation en utilisant le logiciel Echidna et l'analyse de leurs performances ont conduit aux resultats suivants:

- confirmer des comportements prevus par la theorie,
- infirmer certaines previsions theoriques,
- deceler des erreurs dans certains algorithmes et proposer une solution,
- proposer une classification des algorithmes évaluésen fonction de leurs performances,
- enfin, cette etude a permis de mettre en evidence les mecanismes utilisés dans la conception d'algorithmes d'election. Elle pourrait aider a asseoir une methodologie de construction d'algorithmes distribues d'election.

Mots clés: Algorithmes distribues - Election - Simulation - Evaluation - Analyse - Complexite - Performance - Conception d'algorithmes distribues.

ABSTRACT

Because of their complexity, distributed algorithms generate in somme cases, conflicts or differences between theoritical and practical results. Implementation of this type of algorithms may produce unexpected behaviour.

This thesis is devoted to study election algorithms. The problem is fundamental and very elemental in distributed computing. Our aim is to make an inventory of the algorithms, divide them by classes according to the network topology. The algorithms are specified in Estelle and evaluated under Echidna. The analysis results are:

- confirmation of expected behaviours,
- invalidation of theoritical results,
- detection of errors,
- classification of the algorithms,
- and at last, this work brings out the mechanisms and the techiques used to design election algorithms. It may be helpful for establishing a building methodology of leader finding solutions.

Key words: Distributed algorithms - Election - Simulation - Evaluation - Analysis - Complexity - Performance - Distributed algorithms design.

ملخص:

ان الطبيعة المعقدة للخوارزميات الموزعة تؤدي في بعض الحالات الى عدم التناسق بين النتائج النظرية و النتائج التطبيقية . ففي بعض الاحيان يعطي تحقيق هذه الخوارزميات نتائج غير منتظرة أو غير صحيحة . و نظرا لأهمية الخوارزميات الإنتخابية في الحساب الموزّع و استعمالها في تطبيقات عديدة و متنوعة خصصت هذه الأطروحة لدراستها .

و تهدف الدراسة إلى جمع هذه الخوارزميات و تصنيفها الى أقسام حسب شكل الشبكة . و بعد ترجمتها الى لغة برمجة الخوارزميات التوازية ESTELLE و تقييمها عن طريق البرامجية ECHIDNA ، ثم تحليل مدى فعّالياتها ، تم التوصّل إلى النتائج التالية:

- ايجاد نتائج غير معارضة للتنبؤات النظرية ،
- تفنيد بعض هذه النتائج ،
- الكشف عن بعض الاخطا في هذه الخوارزميات و اقتراح حلول لها ،
- اقتراح تصنيف الخوارزميات التي تم تقييمها حسب فعالياتها ،
- وأخيرا ، سمحت هذه الدراسة بإبراز اليكانزمات المستعملة لإنشا الخوارزميات الإنتخابية و قد يساعد ذلك على إقامة منهجية لتصميم هذه الخوارزميات .

الكالمات الدالة :

الخوارزميات الموزعة - انتخاب - تقييم - تحليل - تعقيد - فعالية - تصميم الخوارزميات الموزعة.

S O M M A I R E

CHAPTER 3: ETUDE DES ALGORITHMES D'ELECTION

INTRODUCTION

Ces deux dernieres décennies ont vu croitre la demande en puissance de calcul de telle maniere que la reponse la plus immediate et la moins chère pour les constructeurs a été de multiplier les ressources en calcul. On a vu alors la croissance rapide des réseaux locaux d'ordinateurs a taille reduite et a grands debits de transmission qui ont favorise l'introduction du concept de systeme distribue. Un systeme distribue est un systeme unique dont les composants (données, controle,...) sont distribues sur un ensemble d'equipements informatiques géographiquement repartis et interconnectés par un réseau de communication. Il ne s'agit plus, en effet, d'offrir un moyen de communication et une interface entre les differents systemes eventuellement hétérogènes mais plutöt de realiser un système unique dont les composants sont repartis et cooperent a la realisation d'une tâche commune. En fait, c'est l'association entre l'evolution des réseaux locaux et le developpement de la programmation parallele qui a induit l'apparition d'une nouvelle algorithmique basée sur l'echange de messages entre activités cooperentes. Dans les systemes repartis, ces algorithmes paralleles sont appeles algorithmes distribues.

Dans l'algorithmique distribuee il y a des algorithmes qui realisent generalement les fonctions de contrôle de base inherentes à tout système informatique (exclusion mutuelle par exemple), et des algorithmes qui réalisent des fonctions de controle inherentes a la distribution (detection de la terminaison, l'election, l'interblocage, la gestion de données dupliquées).
Independamment du support sur lequel ils s'exécutent et leur finalité propre, les algorithmes distribues peuvent être définis par l'equivalence suivante [RAY 85]:

alqorithme distribue = processus + messages.

Concevoir un algorithme distribue est un exerci ce difficile, se convaincre de son bon fonctionnement est dans l'etat actuel de l'art plus ardu. En effet, la nature complexe de ce type d'algorithmes (parallélisme, non déterminisme, communication par messages,...) ne permet pas de s'assurer facilement de leur bon fonctionnement. C'est ce constat de la difficult6 de maitrise qui justifie les nombreux travaux d'equipes sur la verification et l'evaluation des algorithmes distribues.
Des outils specifiques ont été conçus pour l'evaluation. Celle-ci permet de mesurer les performances,. elle offre Cgalement un regard original sur certains aspects des algorithmes distribues, et permet d'en mieux saisir les particularités.

Une fonction de controle (le mot controle s'oppose ici a calcul) qui resoud le problème de distinction d'un processus (ou site) parmi un ensemble de processus cooperant a la realisation

d'un but commun, peut prendre deux formes distinctes selon le but recherche [RAY 85]:

- la realisation d'une exclusion mutuelle lorsque le privilege doit être equitablement attribue au processus (et ne peut donc être possédé indefiniment par un même processus).

- l'election d'un processus par les autres processus lorsque le privilege peut Ctre attribue une fois pour toutes.

Cette these est consacrée a l'etude et a l'evaluation des algorithmes distribues traitant le probleme d'election. Elle est organisee en quatre chapitres.

Le chapitre I est un chapitre preliminaire, dans lequel sont présentés principalement, quelques notions de base inhérentes a la distribution, ainsi que des exemples montrant l'utilisation des algorithmes d'election pour resoudre des problemes pouvant se produire dans un système distribué.

Le chapitre II est consacré a l'etude du langaqe de specification de systèmes distribues Estelle [BUD 871, et de l'outil de simulation Echidna [JEZ 89a, JEZ 89bl. Les conditions de simulation y sont décrites.

Le chapitre III comporte l'etude detaillee des alqorithmes d'election pour differentes topologies de reseaux de communication : anneau, complete et quelconque.

Le chapitre IV quant a lui, est reserve à l'analyse des résultats.

Le document est achevé par une conclusion et des annexes.

CHAPITRE 1

NOTIONS FONDAHENTALES DE L'ALGORITHMIQUE DISTRIBUEE

Dans ce chapitre, est donne un aperçu sur l'environnement de conception des algorithmes distribues, puis, sont presentes quelques outils, conçus dans ce domaine, et necessaires a la construction de ce type d'alqorithmes. La nécessité d'avoir des algorithmes d'election est montrée sur des exemples.

I CONTRAINTES DE LA DISTRIBUTION

Le developpement d'un algorithme distribue doit tenir compte des contraintes d'environnement qu'impose la structure materielle sur laquelle il s'execute [COR 81, RAY 85]: l'absence de mémoire commune, l'absence d'un referentiel temporel global, la panne physique des composants materiels du systeme.

1 Absence De Mémoire Commune

La structure materielle d'un systeme distribue est constituée d'un ensemble de machines independantes et communicant a travers des lignes de communication. Il n'y a donc pas de mémoire commune qui puisse servir a la communication et a la synchronisation des composants de l'algorithme ou du systeme distribue. La seule possibilite de communication des composants reste l'échange de messages entre les différents sites du systeme.

Une consequence directe de la communication par messageslproblemes de délai de transmission) est qu'un site donné ne peut connaitre que d'une maniere approchee l'etat d'un autre site. Donc l'état global d'un systeme distribue ne peut être perçu avec exactitude.

2 Absence De Référentiel Temporel Global

La plupart des poblemes de contrôle (de synchronisation en general1 dans un systeme se resolvent par un ordonnancement entre les evenements de celui-ci. En effet, dans une structure distribuee les horloges physiques ne peuvent servir de

referentiel temporel global a cause de leurs derives les unes par rapport aux autres. La notion de temps logique [LAM 781 a été introduite pour la datation et l'ordonnancement entre les Cvenements du système.

3 Non Fiabilité De L'environnement D'exécution

Un des principaux apports des solutions distribuees est de pouvoir maintenir le système en fonctionnement en cas de pannes physiques (portant sur les sites physiques ou sur les moyens de communication) de quelques composantes du système. Un algorithme distribue doit tenir compte de ces problemes.

II QUELQUES CONCEPTS DE BASE DE L'ALGORITHMIQUE DISTRIBUEE

Avec l'algorithmique distribuee sont apparus des téchniques nouvelles: le jeton circulant, l'estampillage, le calcul diffusant et le transfert de connaissances.

1 Le Jeton Circulant

Consiste en une marque particuliere appelee privilege et qui circule sur un anneau de processus. Le processus disposant du privilege entreprend alors une action correspondante au protocole exécuté. Le privilege étant matérialisé par une certaine configuration des variables des processus ou bien par un message special [LEL 771.

2 L'estampillage

En absence d'un référentiel temporel global, l'estampillage reste à la base de la plupart des algotrithmes dans lesquels l'etablissement d'un ordre partiel ou total entre les événements du système est nécessaire. Lamport [LAM 781 a propose le principe de l'estampillage (encore appelé principe des horloges logiquesl qui permet de dater les événements relatifs aux interactions entre les processus (emission/reception de messages] de manière cohérente.

3 Le Calcul Diffusant

Le principe du calcul diffusant a été introduit par Djikstra et Scholter [DIJ 80]. Les processus sont connectés de manière quelconque par les voies de communication, à l'exception d'un processus particulier, qui initialement ne peut qu'émettre des messages. Le principe du calcul diffusant est alors le suivant: Initialement seul ce processus peut. Cmettre des messages, et ensuite tout autre processus ne peut en émettre que s'il en a lui-même reçu. Ce principe est souvant utilisé sur un arbre de recouvrement du graphe des processus. Chacun des processus réexpédie le message reçu vers chacun de ses successeurs et attend la reception des reponses avant d'envoyer a son tour une reponse a son père. Le calcul se termine lorsque le processus racine a reçu les réponses de tous ses fils.

4 Le Transfert De Connaissances

Les processus constituant un algorithme distribue, bien qu'ils coopérent à la realisation d'une tâche commune ne disposent initialement que d'informations locales (généralement des informations sur les voisins). Pour une prise de decision, un processus a besoin d'acquerir des informations qu'ont les autres processus afin de constituer une vue partielle ou globale sur l'état du systeme. C'est le principe du transfert des connaissances introduit dans [HEL 86al.

La plupart des protocoles de transfert des connaissances se basent sur la technique de la vague. Un processus desirant acquérir des informations sur le systeme diffuse a un processus voisin un message approprie. Ce message sera enrichi, a sa reception, des connaissances locales du site recepteur et diffuse aux voisins de ce dernier, jusqu'a atteindre l'ensemble des processus presents dans le systeme. Le risque de multiplication excessive des messaqes de transfert des connaissances peut être évité par une technique de contrôle de ce type de transferts [HEL 86a].

III LES QUALITES D'UN ALGORITHHE DISTRIBUE

Les qualités d'un algorithme distribué se mesurent par les criteres suivants [RAY 85]:

1 Hypotheses Sur Le Réseau De Communication

Un algorithme distribue pose toujours des hypotheses sur le maillage du réseau d e communication d ' une part, et sur le comportement des voies de communication d'autre part.

a) Mâillâge du réseau de communication

La plupart des algorithmes distribues supposent une topologie particuliere, peu d'entre eux s'executent sur un reseau quelconque. Les plus tout-antes sont:

- Maillage en anneau: un processus ne connait que ses deux voisins immediats (gauche et droitl. L'anneau forme par les processus prut etre uni- ou bi-directionnel selon que les messages circulent dans un seul sens ou dans deux sens.

- Maillage en arbre: structure hierarchique ou a tout noeud est associe un processus qui ne peut communiquer qu'avec son pere, et le processus racine ne communique qu'avec ses fils.

- Maillaqe complet: tout processus peut communiquer avec tout autre processus.

Une topologie quelconque est une structure mixte des topologies précédentes.

b) Propriétés comportementales des liaisons

La plupart des algorithmes distribues font des hypotheses sur le comportement des voies de communication. Les plus frequement rencontrées s'ennocent comme suit:

- Il n'y a pas de duplication de messages au cours de la transmission.

- Les messages ne sont pas alter-es.

- 11 n'y a pas de dessequencement.

- Le délai d'acheminement des messages bien qu'aleatoire, il est fini; tout message est reçu au bout d'un temps fini. Il n'y a pas de perte de messages.

2 Degré De R&partition (Symétrie De L'algorithme)

Le degré de repartition permet de definir les niveaux de symetrie des rôles joués par les differents processus du systeme. Dans [RAY 85], on distingue plusieurs niveau de symetrie. Il sont présentés ici par ordre croissant:

- Non-symetrie: les processus n'exécutent pas tous le même texte (exemple du systeme client-serveur).

- Symetrie de texte: les processus executent tous le même texte, sauf que l'identite du processus apparait dans le texte. Ceci peut conduire à des comportements differents.

- Symetrie forte: les processus executent tous le même texte sans aucune reference à leurs identites. Selon les messages reçus, ils peuvent avoir des comportements identiques ou differents.

- Symetrie totale: les processus executent tous le même texte et engendrent des comportements identiques.

3 Resistance Aux Pannes

L'algorithme distribue est d'autant plus interessant qu'il resiste aux pannes. Ce critère est lie au precedent vu que la panne d'un site dans un algorithme non symetrique risque d'être fatale à l'algorithme.

4 Etat Global Ou Local

Un alqorithme est plus performant si pour un processus de l'algorithme, la prise de decisions n'utilise pas un etat global. Outre la reduction du nombre de messages echanqes cela

peut permettre d'assurer une meilleur resistance aux pannes.

5 Complexité En Trafic Et En Temps

Pour une même fonction réalisée, un algorithme distribue est d'autant plus interessant qu'il minimise le nombre de messages(trafic) engendrés sur le rdseau. Le temps necessaire a la realisation d'une operation de controle est un autre critere de mesure de performance d'un algorithme distribue (temps d'electionl.

IV LES ALGORITHHES D'ELECTION

L'élection est utilisee dans un systeme reparti quand un site est appelé a jouer un role de supervision, ou quand un site, sans être le coordinateur de l'ensemble des sites, joue un role particulier et que la panne de ce site est un événement auquel l'application doit reagir. Les applications qui utilisent ce mecanisme sont très nombreuses. Dans ce qui suit, sont donnés en bref, quelques exemples d'utilisation de ce type d'algorithmes.

Exemple 1:

C'est par exemple le cas dans les systemes teleinformatiques, où la gestion du reseau de communication nécessite un ensemble de protocoles assurant un service fiable et de bonne qualite. Parmi ses fonctions, la couche réseau (3º niveau de la structure de réseau definie par la norme ISO) consiste a sélectionner le chemin que doit suivre un message entre deux adresses. Le r-outage est un des elements primordiaux qui permet la reconfiguration du systeme en cas de panne, d'anomalie ou tout simplement de congestion. Parmi les techniques de routage, il y a celles qui sont caracterisees par un centre de controle [PUJ 871 dont le rôle consiste a "élaborer" des tables de routage de façon statique ou dynamique. La défaillance du noeud de controle nécessite la designation d'un autre noeud pour assurer cette fonction.

Exemple 2:

De même, dans certains reseaux locaux en anneau [PUJ 871 une station a un rôle important pour l'adminstration de l'anneau(contrôle d'accès aux ressources de communication, recuperation des messages perdus, etc..). C'est le cas par exemple, du reseau local "token ring" d'IBM. Cette politique est decrite par la norme IEEE 802.5. Lorsqu'une station est arrêtée ou est en panne, elle est automatiquement retiree de l'anneau. Si c'est l'administrateur, il faut en elire un nouveau.

Exemple 3:

Certaines applications necessitent l'utilisation de protocoles de diffusion fiable. Les protocoles de diffusion sont des algorithmes distribues qui permettent l'envoi d'un même message à un ensemble de processus destinataires en une seule operation. L'ordre de reception des messages au niveau des sites peut être quelconque, FIFO, global, causal ou atomique.

Deux protocoles de diffusion caracterises par un ordonnancement global sont donnés dans [CHA 841 et [NAV 881. L'ordonnancement global assure que les messages sont reçus dans le même ordre par tous les sites recepteurs. Ceci peut être utilisé, par exemple, pour simplifier la conception de mecanismes du contrôle de concurrence et des procedures de recouvrement dans les systèmes de bases de donnees distribues.

Le protocole de diffusion ordonnee propose par Chang et Maxemchuk [CHA 841 est base sur un site particulier appele “site jeton". Ce dernier joue le role d'un entonoire par lequel transitent tous les messages diffuses. Le site jeton est charge d'assurer que les messages sont tous reçus, et, dans le même ordre par les destinataires. La fiabilite de l'algorithme est assuree en attribuant, de façon circulaire, ce privilege a un ensemble de sites qui constituent la "liste jeton". Lorsqu'un site détecte une defaillance, il invite les autres sites a reconstituer une nouvelle "liste jeton" et a élire un nouveau "site jeton".

Exemple 4:

Dans les bases de donnees réparties, le problème du contrôle de la concurrence, dû a l'execution parallele de transactions accedant a des donnees dupliquees, peut Ptre résolu en associant à chaque ensemble de donnees (dupliquées sur les divers sites du réseau) un site particulier. Le rôle de ce site, appele site primaire [BER 81] est de contrôler les accés qu’effectuent les transactions de façon a assurer la coherence mutuelle de ces donnees (respect des contraintes d'intégrité) et la convergence de leurs copies vers une même valeur.

Exemple 5:

Un autre exemple est celui de l’algorithme distribué de synchronisation d’horloges TEMPO conçu pour synchroniser les horloges du système UNIX distribué de Berkeley. Tempo est base sur une structure maitre-esclave composée de processus (time deamons) s'executant sur un réseau local . de machines individuelles; un processus par site. Un processus maitre (master time deamonl mesure la difference entre l’heure indiquée par l' horloge locale de son site et celles indiquees par tous les autres sites. Le maitre calcule l’heure globale du réseau comme étant la moyenne des valeurs fournies par toutes les autres horloges(non defaillantesl du reseau. Il transmet ensuite a chaque processus esclave la valeur avec laquelle ce

dernier corrigera l'horloge locale a son site de residence.

Ce processus de synchronisation d'horloge est répété periodiquement. Une description detaillee de Tempo est donnee dans [GUS 84, GUS 851. Pour s'assurer que Tempo fournit un service continu et fiable, il est necessaire d'implementer un algorithme d'election [GUS 861 qui elira un nouveau maitre si le site sur lequel s'execute le processus maitre courant tombe en panne, ou si le reseau des processus est partitionne en sous-réseau.

Dans ces systemes, la panne du site jouant un role particulier nécessite l'election d'un nouveau site pour le remplacement du site defaillant et le maintien du systeme en fonctionnement.

Exemple 6:

Un autre cas necessitant l'election d'un site parmi les autres, est celui de plusieurs sites d'un reseau cooperant a la realisation d'une tâche commune, où un des composants tombe en panne [GAR 821. 11 peut être alors necessaire de reconfigurer le systeme afin, soit de poursuivre la tâche en question, soit de selectionner de nouvelles tâches qu'il faudrait assigner aux differents sites. La reconfiguration du systeme doit être a la charge d'un site unique, sinon il est possible d'aboutir à une confusion. Donc, pour assurer la reconfiguration correcte du systeme, il faut élire un site coordinateur qui s'en chargera.

CHAPITRE 2

EVALUATION SOUS Echidna

I MOTIVATIONS ET OBJECTIFS

Le processus de developpement d'un alqorithme distribue passe toujours par l'étape de validation-evaluation de celui-ci. Cette validation-evaluation se fait généralement par une preuve formelle des propriétés de l'algorithme et une analyse theorique de complexité (trafic, tempsl et ensuite par une simulation de l'algorithme sur une machine centralisee.

La simulation [GRO 871 est une technique largement utilisee dans tous les domaines lorsque la complexite d'un probleme rend les tentatives de solution analytique inoperantes. La verification d'alqorithmes distribués même simples rentre dans ce cadre du fait de l'explosion combinatoire des comportements (ensemble infini de comportementsl de ces alqorithmes.
Toutes les méthodes de validation automatique ont leurs limites; la validation est toujours reduite a une partie du comportement, et la partie prise en compte par les simulations est souvent plus importante que les parties prises en compte avec d'autres methodes. La simulation d'algorithmes distribués peut aussi aider a la conception de ce type d'algorithmes.
En effet la simulation peut traiter des modèles d'algorithmes assez complets et permet de détecter efficacement des erreurs éventuelles sur un sous-ensemble des comportements possibles du protocole. Elle permet de completer la preuve formelle, et de tirer des conclusions plus générales quant au comportement de l'algorithme distribué reel. En effet, l'implantation d'un prototype dans un environement d'exécution réel n'est pas toujours possible et de toute façon les conclusions qui peuvent en Ptre tirées dependent etroitement de la mise en oeuvre [ADA 881.

Un grand nombre d'algorithmes distribués d'élection ont été publiés pour diverses topologies de réseaux de communication et avec divers concepts de base.
11 serait interessant de faire un inventaire de ces algorithmes en les repartissant par classe en fonction de la topologie du réseau, et de les implementer (simuler) et d'analyser leurs performances afin de les comparer.

Les resultats escomptes sont:
- trouver les meilleurs algorithmes,
- confirmer ou infirmer les previsions theoriques quant a leur bon fonctionnement et leur performance,
- mettre en evidence des comportements eventuels, non prévus par la théorie,

L'evaluation des algorithmes est réalisée a l'aide du logiciel Echidna [JEZ 89a, JEZ 89b]. C'est un système conçu dans le but de fournir les outils necessaires à l'experimentation et/ou la simulation d'algorithmes distribues. Echidna est construit autour d'un compilateur d'algorithmes distribues décrits dans le langage de specification Estelle CISO 891. Echidna permet principalement la simulation du parallélisme.

II LES OUTILS D'EVALUATION

1 Le Logiciel

l'evaluation consiste a specifier les algorithmes étudiés en langage Estelle de specification de systemes distribues. Ces algorithmes sont ensuite implémentés(simulés) sous le systeme Echidna qui a été développé sur un hypercube IPSC/2 d'INTEL, sur des reseaux de stations SUN, et sur les micro-ordinateurs PC.

1.1 Langage de spécification Estellg

La difficulté a prévoir toutes les configurations possibles lors de la conception d'un algorithme distribue, l'ambiguité, la longueur, la lourdeur et les differences d'interpretation d'une description en langage nature1 montrent l'interêt a specifier les algorithmes distribues dans un langage de description formelle [JAR 871. Pour cela, de nombreux modeles de description des algorithmes (ou des protocolesl ont été développés de façon a pallier aux problemes ci-dessus. Ils peuvent être classes en trois grands types de techniques:

1. Les systemes a transitions, c'est-a-dire d'une part les automates d'etats finis et d'autre part, les modèles de graphes à marquage comme les réseaux de Petri [RAZ 84, JAR 85];

2. Les langages de haut niveau sequentiels qui ont été plus ou moins adaptes au domaine par la possibilite d'exprimer l'arrivée d'événements externes [BOU 881.

3. Les techniques algebriques et logiques [AUD 881.

Estelle est un langage basé sur le modele de transitions d'etats etendu par le lanqage Pascal avec certaines restrictions plus des elements d'expression des comportements paralleles. Une specification Estelle est la description du comportement

d'un systeme fermé; c'est-a-dire sans interaction avec l'extérieur.

Tous les objets Estelle sont fortement types; la notion de typage Pascal est etendue aux objets paralleles. On parle alors d'un type de canal, de comportement et d'un type de module.

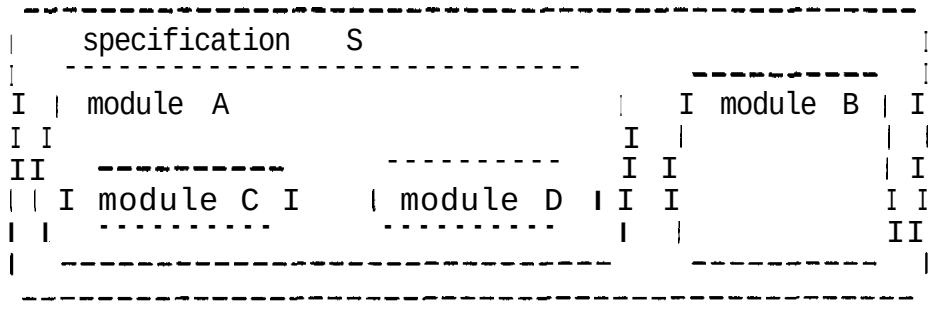
Dans le paragraphe suivant sont présentés les concepts fondamentaux d' Estelle. Une description plus détaillée d'Estelle est donnée dans [COU 86, BUD 87, IS0 891.

a) Les concepts de base

Ce paragraphe presente les concepts de base specifiques du langage Estelle (expression des comportemants parallèles). 11 s'agit du sous-ensemble Estelle implémenté sous Echidna.

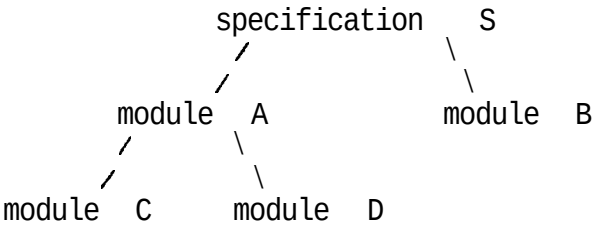
b) Notion de module

Une specification Estelle est une collection de composants communicants appelés instances de modules.



Un module est defini dans la specification par:
- une interface de communication avec son environnement,
- et, par son comportement a la definition d'un entête(header), et d'un corps(body).

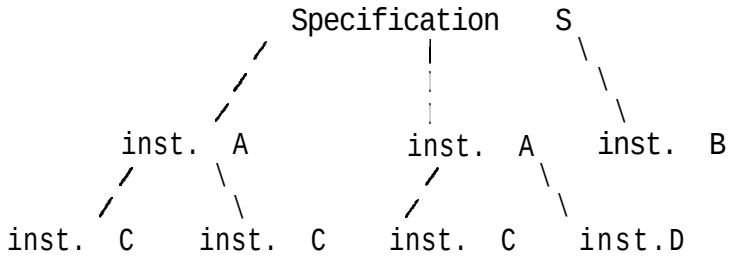
La definition d'un module peut comprendre des definitions d'autres modules (modules fils). Des definitions de modules en structure d'arbre hiérarchique sont ainsi obtenues, la racine(module principale) representant toute la specification. 11 y a toujours une instance unique du module principal. On peut avoir par exemple, l'arbre des definitions de modules suivant:



L' entete d'un module definit l'interface du module avec son environnement extérieur. En d'autres termes, il décrit sa visibilite extérieure. Une instance de module est l'execution d'un module. 11 y a tou jours une instance unique de la specification qui est un module particulier. Une instance de module ne peut être créée que par une instance du module enqlobant. Ceci fait que l'arbre des definitions est un modele pour l'arbre hiérarchique des instances de modules.

Exemple:

Par exemple, pour la specification S, un arbre d'execution . possible est l'arbre suivant avec deux instances du module A et une instance du module B. La premiere instance de A a initialise deux instances C, alors que la deuxieme a lance une instance C et une instance D.



L'entête d'un module definit sa visibilite exterieure sous la forme d'un ensemble de points d'interaction, et de parametres.

```
Module  type-entetel (<paramètres>);
  ip       <ensemble de points d'intéraction>
End;
```

Pour un même entete de module il peut y avoir differents comportements internes et sous-structures. A un corps(body), est toujours associe un entete.

Exemple: Body type B for type-entetel;

Le comportement interne du module est donne par un modele de transition d'etats defini par:

. un ensemble d'etats de contrôle,
. et, un ensemble de transitions.

Syntaxiquement, le corps d'un module est compose de :
- une partie declarations
- une partie initialisations;
- et, une partie transitions.

```
Body a For b;
  <declarations>
  ‹initialisations›
  ‹transitions›
end;
```

1. Partie declarations:

La partie declarations comprend les declarations Pascal usuelles(constantes, types, variables, procedures et fonctionsl et des declarations d'objets Estelle: canaux, entêtes et corps de modules, états de contrôle et variables modules.

La description d'un canal contient une enumeration des interactions qui peuvent circuler sur celui-ci dans un sens ou dans l'autre. Syntaxiquement, la definition d'un canal est donnée comme suit:

```
Channel nom-type-canal (id-role-extl, id-role-ext21;
    By id-role-extl:tdeclaration d'intéractions›;
    By id-role-ext2:tdeclaration d'intéractions›;
```

Exemple:

```
Channel canal(rôle1,rôle2);
   By rôle1 : M1;...,Ml;
   By role2  : N1,...,Np;
   By rôle1, sortie2 : K1,...,Kq;
```

La declaration d'une interaction comporte le nom de l'interaction et eventuellement une liste de parametres types.

Exemple: Requête(X:integer; Y:boolean);

Deux points d'interaction P1 et P2 peuvent etre declares comme suit:

```
Ip    P1 : canal(rôle2);
      P2 : canal(rôle1);
```

P1 utilise le canal de communication de type canal et joue le rôle rôle1. P1 peut emettre tous les messages Mi et Kj, et recevoir les messages Ns. P2 peut emettre Ns, et recevoir Kj et Mi.

Exemple :

```
Channel canal1(rôle1, rôle2);
  By rôle1 : Ml;... Mn;
  By role  : ;
```

Cet exemple montre la declaration d'un canal unidirectionnel ou seule l'extrémité qui jouera le rôle "rôle1" pourra emettre des interactions.

A chaque point d'interaction est associe une file d'attente non bornée de discipline FIFO.

Exemple de declaration de coprs de module:

```
Body Al for A;

  Const tconstantes Pascal>;
  Type  <declaration de types Pascal>;
  Var   <   "       de variables Pascal>;

  Channel canal(rôle1, rôle2);
    By rôle1 : request(X:integer; Y:boolean);
    By role2 : response;

  State   active, wait;
  stateset   aw-[active, wait];

  Module B(moi: integer);
    I   ip   P1: canal (rôle1);
    |        P2 : canal(rôle2);
  End;

  Body  Bl For B;
    I .
    | .
  End;

  Modvar  int1, int2 : b;
end ;
```

Les variables de types modules introduites par le mot-clé Modvad servent de reference aux instances de module d'un certain type d'entete. Elles permettent de designer les instances.

2. Partie initialisation:

La partie initialisation spécifie les valeurs de certaines variables avec lesquelles sera créée chaque instance de module. L'initialisation d'une variable module signifie la creation d'un module fils. Elle etablit aussi les liens de communication entre points d'interaction.

Exemple:

```
Initialize
   TO état-initial (état initial du module)
   Begin
     ‹initialisations›
     ‹établissement des liens de communication>
   End;
```

3. Partie transitions:

La partie transition est un ensemble de declarations de transitions. La definition d'une transition est constituée d'une liste de gardes qui exprime la condition de tir de la transition, d'un bloc comportant des declarations Pascal et d'une sequence d'instructions Pascal et Estelle constituant la partie action de la transition.

Syntaxe d'une transition

```
Trans
     <condition>
     {declarations locales>
     <action>
```

La condition d'une transition est exprimde a l'aide des clauses suivantes:

- .FROM : définit l'état de depart;
- .WHEN : exprime une attente d'un message;
- .PRIORITY : assigne une priority a la transition, elle servira lors du choix des transitions tirables;
- .PROVIDED : exprime la satisfaction d' une expression booléenne;

c) Les instructions Estelle.

1. Creation d'une instance:

L'initialisation d'une instance se fait grâce à l'instruction INIT:

INIT idf-module-variable WITH idf-body(‹liste par effectifs›)

La variable "module-var" dans l'instruction "INIT" fait reference a la nouvelle creation d'instance.de module. 11 peut y avoir plusieurs corps de modules associes a la même entête de module. L'instruction "INIT" différencie entre eux en specifiant "idf-body".

2. Connexion de points d'interaction

La connexion des points d'interaction se fait grâce à l'instruction CONNECT.

Exemple: Connect P1 TO P2

Cette demande de connexion ne peut Ptre satisfaite que si:
. les deux points d'interaction font reference au même canal et jouent des roles opposes ,
. aucun des deux points d'interaction references n'a deja été connecté.
Une interaction ne peut être Cmise qu'à travers deux points d'interaction connect&s.

3. Emission d'une interaction

La file d'attente associée à un point d'interaction est mise a jour lors des emissions et receptions de messages. En Estelle il est possible d'emettre un message grace a l'instruction OUTPUT qui spécifie le message attendu et le point d'interaction sur lequel il est attendu.

Exemple: OUTPUT P1.m

4. Reception d'une interaction

La clause WHEN d'une garde permet la reception d'une interaction en deux étapes:

- verification de l'existance en tête de file du message attendu pour l'evaluation de la garde,
- consommation du message, si la transition est tiree.

5. des restrictions au langage Pascal sont admises, par exemple, il n'y a pas d'instructions de manipulation de fichiers.

Dans l'annexe est présentée une specification Estelle de l'algorithme de Chang et Roberts [CHA 791.

1.2 Presentation D'Echidna

Echidna [JAR 89, JEZ 89] est construit autour d'un compilateur d'algorithmes distribués décrits en Estelle et fournissant du code pour machines multi-processeurs et d'un ensemble d'outils pour observer le comportement de. l'algorithme distribue sous experimentation.

L'execution de la specification Estelle va donc, consister a executer en parallele les tâches sur chaque site. Pour mettre en oeuvre ce modèle d'execution, le compilateur traduit la description formelle Estelle en un ensemble de structures de données, exprimees en langage C, qui est ensuite compile par le

compilateur C local, puis lie avec le noyau d'execution distribue (NED) avant d'être charge et exécuté sur le système cible.

Le NED a la charge de connaitre et de gérer la communication non locale entre tdches Estelle. Il est totalement ecrit en C, et découpé en trois modules:

1. L'exécuteur:

   Un modèle executable est obtenu après edition de lien entre le NED et le code généré par le compilateur. 11 est charge sur tous les sites. Parmi les fonctions de l'executeur on distingue:
   - l'analyse des commandes,
   - la creation de la tâche englobante (racine de l'arbre) et son initialisation en exécutant sa "partie initialisation".
   - le lancement du scheduler. Le scheduler assure l'enchaÿnement des tâches initialisées sur chaque noeud du réseau. Le code du scheduler est ecrit en C.

2. Le runtime Estelle:

   Il a pour rôle de reconstruire une machine virtuelle permettant l'execution d'un programme Estelle a partir du runtime C qui existe autour de la machine. 11 doit fournir les primitives de manipulation d'objets Estelle et Pascal qui n'existent pas en C, un générateur aleatoire (pour le choix non deterministe parmi les transitions tirables) et reconnaitre la communication distante pour l'acheminer par le réseau.

3. L'interface avec le système:

   Le but est de fournir de maniere homogene l'abstraction "station d'experimentation" quelque soit le système distribue sous-jacent. Il fournit des primitives permettant d' emettre des messages vers les autres sites et de tamponner les messages qui arrivent des autres sites. 11 fournit également une horloge locale, gère les entées/sorties et l'identificateur de site. Seule cette partie du NED est a adapter à chaque nouveau système.

2 Le Matériel

Le travail présenté dans cette these a été réalisé sur la version V0.2(1989) d'Echidna implémentée sur un micro-ordinteur du type PC sous le systeme d'exploitation MS/DOS.

III HISE EN OEUVRE DE LA SIMULATION SOUS Echidna

Ce paraqraphe presente les conditions d'evaluation ainsi que les parametres pris en compte.

1 Le Modèle D'election

Deux modèles sont prevus:

- Election collective: l'ensemble des sites presents dans le systeme lancent, en parallele, une operation d'election.

- Election individuelle: un site particulier du systeme initialise l'operation d'election.

2 Les Critères D'évaluation

Deux criteres de performances ont été mesurés:

1. **Complexité** en nombre de **messages(trafic):**

   c'est le nombre de messages échangés, necessaires a la determination de l'identite de l'elu.

2. **Complexité** en temps **d'élection:**

   C'est la durée necessaire a la determination de l'identite du site Clu.

   Le calcul d'une election se termine dans un seul site qui se chargera alors d'en informer les autres sites. Les complexités évaluées dans ce document ne tiennent pas compte de cette phase de diffusion de l'identite de l'élu, car, elle ne fait pas toujours partie du protocole d'election.

3 Le Modèle Estelle Implanté Sous Echidna

Une description Estelle est un arbre dont les noeuds sont des tâches et la racine la specification englobante. Une tâche est composée d'une specification (module] et d'un corps (body). Un module est crée lors de sa specification dans le module immediatement englobant, et son corps lui est attribué lors de l'initialisation de la tâche. Les fils d'une tâche sont les instances des modules(variables de type module).

Sous Echidna, seul un sous-ensemble "statique" d'Estelle [JEZ 891, qui permet toujours la creation dynamique de l'arbre, mais pas sa modification a été implanté. En effet, les tâches non terminales sont passives: donc il ne reste sur chaque site que l'ensemble des tâches terminales créées et placées dynamiquement sur ce site. L'execution du proqramme Estelle va donc consister, a partir de la, a executer en parallele les tâches sur chaque site. Cette implementation d'Estelle est suffisante pour la specification et la mise en oeuvre des

algorithmes de contrôle distribues.

Deux niveaux de parallelisme sont mis en evidence: le parallelisme inter-noeuds qui est le vrai parallelisme, et le parallelisme intra-noeuds ou quasi-parallelisme. Ce parallelisme est simulé sur chaque site par un scheduler dans le cas où plusieurs tâches Estelle sont lancées sur un même site physique. L'utilisiateur a, cependant, la possibilite de choisir de simuler l'asynchronisme entre plusieurs processus d'un même site.

4 La Simulation Sous Echidna

Une machine centralisee est un cas simple de systeme reparti, qui est traité, sous Echidna, comme une machine parallele ne disposant que d'un site physique. Dans ce cas, Echidna se présente comme un simulateur Estelle pouvant fonctionner sur de petites machines (type PC) sous les systtmes UNIX ou MS/DOS (cas de cette thèse).

La figure II.1 represente la structure d'un algorithme d'election.

Un algorithme d'election est un algorithme distribue compose d'un ensemble de processus s'executant chacun sur un site, et communiquant par échange de messages.

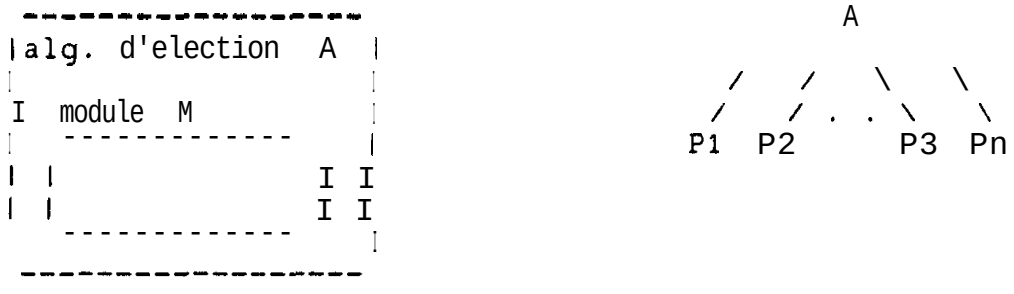


Figure 11.1: Modèle d'un algorithme d'élection

L'ensemble {p1, p2, . . .pn} des instances de M sont des images identiques exécutables d'un même modèle de comportement. En effet, les algorithmes étudiés presentent tous une symetrie de texte. Le systeme est donc compose d'un seul site physique et de n sites logiques (processus).

Les comportements des algorithmes sont observés sous les deux modes d'exécution synchrone et asynchrone offerts par Echidna.

5 Collecte Des Informations De Mesure

Dans le système Echidna l'instruction "Trace" [JEZ 89a, JEZ 89b] est la seule extension au langage Estelle (qui ne possede pas d'instructions d'Entrées/Sortiés) qui a été faite. Sa syntaxe est celle d'un writeln Pascal, mais sa semantique est propre au contexte de l'expérimentation. En effet, pour ne pas perturber l'experimentatibn, une option prévue dans la commande de lancement d'execution permet de forcer l'instruction "Trace" a effectuer les Entrees/Sorties en memoire afin d'être transferees physiquement seulement après que 1'experience soit terminee.

L'instruction "Trace" a été utilisee pour recueillir les informations désirées (nombre de messages et temps d'élection) et qui sont enregistrées a u cours de l'execution de l'algorithme.

11 faut noter également que l'evaluation des algorithmes a été faite independamment de toute application.

Le schema de la figure II.2 résume les differentes etapes par lesquelles transite un algorithme exécuté sous Echidna.

a) Evaluation du temps

Une des caracteristiques des systemes distribues est l'absence d'un référentiel temporel global. Afin de pallier a cet inconvenient, de nombreux travaux ont été menes sur ce problème qui consiste à réaliser une horloge(logique) globale fournissant un temps abstrait global, servant de référentiel commun a tous les sites[LAM 78, JUD 87, JEZ 98a]. Cette perception du temps, est necessaire essentiellement pour resoudre les problèmes de synchronisation(mécanismes d'allocation de ressources telle que l'exlusion mutuelle, la coordination de processus, les relations de precedence causale entre evenements, etc).

Dans cette these, il est question de mesurer le temps d'execution des algorithmes. Pour être pratique, il n'est pas necessaire d'essayer de mesurer le temps d'execution "reel" d'un algorithme. Ceci pour deux raisons:

- D'une part, deux executions differentes aboutissent a des résul tats differents, car elles sont liées a l'environnement d'execution et a la charge du système qui varient d'un instant a l'autre.

- D'autre part, les resultats obtenus vont dependre de la mise en oeuvre. C'est-a-dire, deux implementations du même algorithme donneraient des performances differentes, si elles sont réalisées sur des systèmes differents.

Dans ces conditions, la comparaison des performances d'algorithmes distincts (c'est la le but de notre travail) pourrait aboutir à des resultats specifiques, voire non

siqnificatifs. En effet, les conclusions qui peuvent en être tirees ne seront valables que dans un contexte equivalent a celui de l'evaluation. Avec des mises en oeuvre differentes il est possible d'obtenir des conclusions completement differentes et même contradictoires.

Donc, pour realiser l'evaluation du temps, un modèle base sur le fonctionnement du systeme ECHIDNA(scheduler) a été defini.

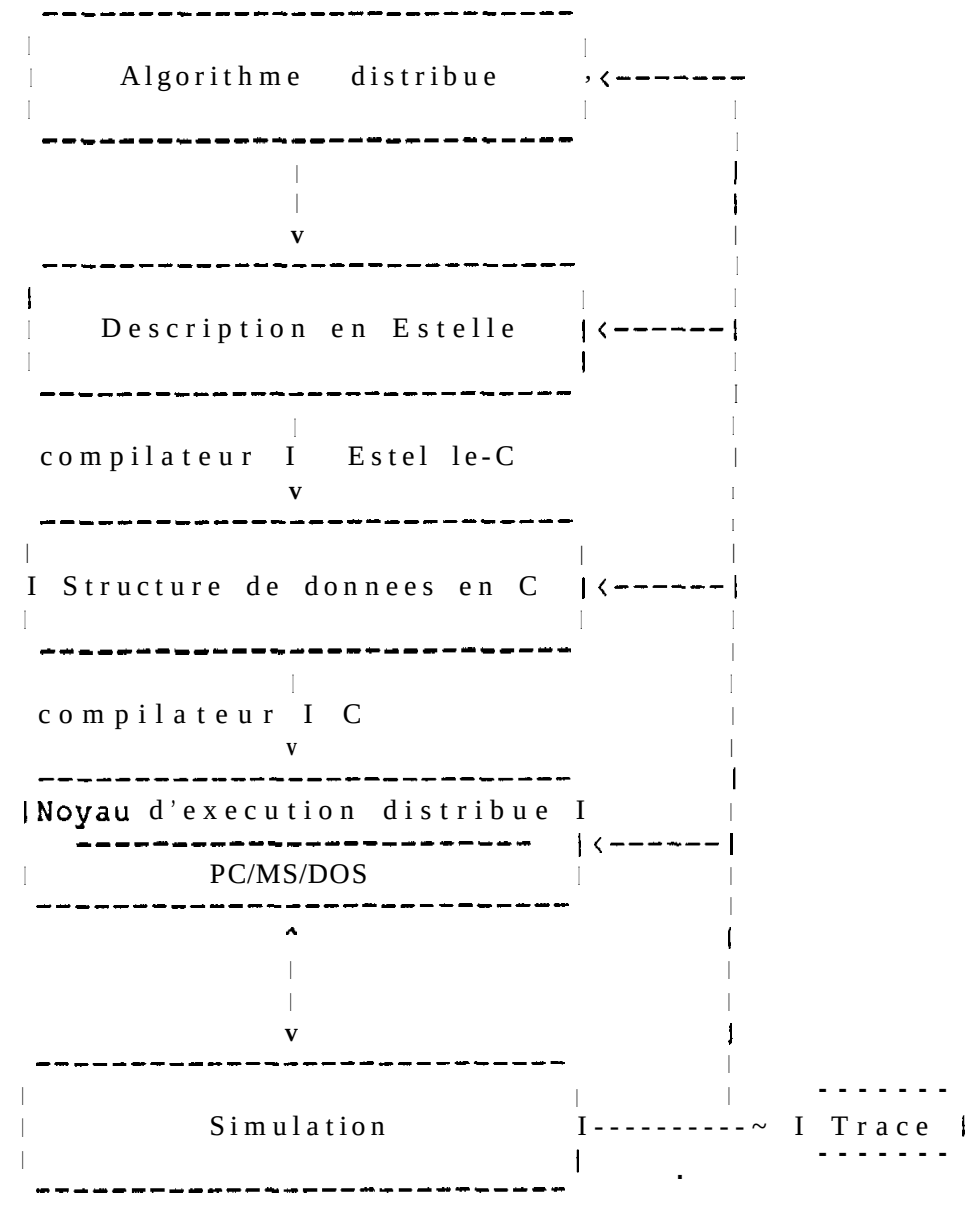
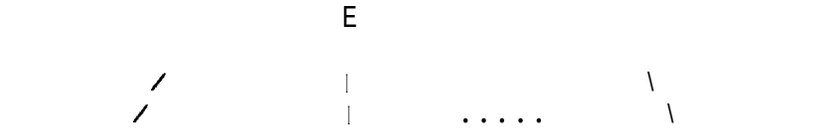


Figure 11.2: Simulation d'algorithme distribué sous Echidna

b) Le principe d'évaluation:

Soit un algorithme **E** compose d'un ensemble de n processus. Un processus est décrit par un ensemble de transitions [transij], i=1..m, et j=1..n.



trans 11	trans 12	trans 1n
trans 21	trans 22	trans 2n
trans il	trans i2	trans in
trans ml	trans m2	trans mn

Le scheduler est charge d'assurer l'enchainement de l'executions des tâches initialisées sur chaque noeud. Ainsi le scheduler effectue sur chaque site l'alqorithme simplifié suivant:

```
Répéter jusqu'à fin experimentation
   lrecueillir les messages arrivant de l'exterieur
   |                                    (s'il y en al;
   lpour chaque processus du site
   |       choisir une transition tirable.
   lpour chaque processus du site
   |(ou seulement pour un sous-ensemble d'entre eux si
   |  l'asynchronisme est simulé)
   |       tirer la transition choisie de manière
   |       atomique;
Fin répéter
```

Il est clair que dans l'algorithme ci-dessus, le scheduler travaille par étapes de calcul, dans chacune d'elles, il execute une sequence de transitions. En effet, pour n processus lances:

- n transitions sont exécutées en une étape de calcul dans le cas d'une execution parallele synchrone.

- t<=n transitions sont exécutées dans le cas d'une execution parallele asynchrone.

Examinons de près l'execution de ll'algorithme E qui est supposee s'effectuer en trois Ctapes:

étape1:{ trans21, trans12, transmn}
étape2:{ trans31, trans42, trans2n}
étape3:{ trans72, transk6n}

1. Durant chacune des k étapes (k=1 a 31, tout message émis sera reçu durant la même etape. 11 pourra(selon que la transition correspondante soit exécutée ou non) être traite dans l'étape suivante, car il s'agit d'un système centralisé.

2. On suppose que la durée d'execution d'une etape represente une unite de temps.

L'evaluation du temps d'éxécution d'un algorithme peut donc être représentée par le nombre d'étapes de calcul qui ont été effectuees. Pour l'algorithme E ci-dessus, le temps mesuré sera égal a 3 unites.

CHAPITRE 3

ETUDE DES ALGORITHMES D'ELECTION

I DEFINITIONS ET TERMINOLOGIE

1 Concept De Processus

Un processus est considéré comme une unite d'execution élémentaire d'un algorithme distribue ou parallele. Plusieurs processus peuvent operer simultanement. Dans un algorithme distribue, un processus est considéré comme un algorithme séquentiel(au point de vue de son execution) deterministe ou non, muni de primitives de communication lui permettant d'envoyer et de recevoir des messages, afin de communiquer avec les autres processus.

2 Réseau Anonyme

C'est un reseau dans lequel les processus ne sont pas dotes de numeros d'identifications tous differents, ou encore qui ont tous la même identite [LAV 87, MAT 891.

3 Modélisation Du Réseau

Le reseau de communication est modelise par un graphe G=(V,E) connexe. Les noeuds (V) et les aretes (E) representent respectivement les processus et les canaux de communication.

4 Communication Synchrone Et Asynchrone:

Dans le mode de communication synchrone, une horloge globale est connectée a tous les sites du reseau. L'intervalle de temps entre deux pulsations consécutives de l'horloge est le "round". Au debut de chaque round, chaque site decide, en se basant sur son état, quels messages envoyer et vers quelle destination. Chaque site recevant des messages durant ce round, les utilise ainsi que son contexte local pour decider de l'action a entreprendre. Les sites reveilles spontanement commencent l'algorithme en entrant dans un état initial, et en attendant le debut du prochain round.
Dans le mode asynchrone il n'y a pas d'horloge globale, et les délais d'acheminement des messages sont variables [AFE 85, FRE 85, ADA 881.

II LANGAGE DE DESCRIPTION ET NOTATIONS

Le texte de chaque alqorithme est exprime par la suite à l'aide d'un langage de type "événementiel". A chaque evenement ei est associée une action ai ddcrivant le comportement d'un processus. Un tel Cvenement est soit l'émission ou la reception d'un message donné, soit la validation d'une expression booléenne.

L'emission d'un message se fait par la primitive:
envoyer‹type de message›(message) à destination.

La notation suivante est utilisée:
∀ i ∈ V : envoyer‹type de message›(msg) à Pi;
signifie envoyer à tous les Pi, tel que i soit dans l'ensemble V, le message msg.

La reception d'un message se fait par la primitive:
recevoir ‹type message›(message) depuis origine.

La reception d'un message par Pi depuis Pj provoque une interruption de la façon suivante:
Lors de la reception ‹type message›(message) depuis j
faire
-- le traitement du message --
fait.

Les notations suivantes sont utilisees tout le long de cette these:

. env-vg: envoyer au voisin gauche,
. env-vd: envoyer au voisin droit,
. faire-suivre: env-vg le message reçu par la droite, et env-vd le message reçu par la gauche.
. IDi: l'identité du processus Pi,
. Pmax: le processus d'identite maximale,
. n:le nombre de processus,
. log x: logarithme decimal de x,
. lg x: logarithme népérien de x,
. Log x: logarithme en base 2 de x,
. Hn=0.693..Log n.

III PRESENTATION D'UN ECHANTILLON D'ALGORITHHES

Le problème de l'election a été étudié de façon intensive ces dernieres années sous l'hypothese d'identites distinctes des sites du réseau. 11 s'agit alors d'un probleme de calcul d'extremums, et divers algorithmes ont été publiés dans differents contextes et sous des hypotheses multiples:

- selon la topologie du réseau (anneau ou maillage complet le plus souvent);

- selon l'information sur la topologie globale du réseau qui est

mise a la disposition des sites (connaissance de la topologie particuliere du reseau, ou seulement de sa taille, connaissance des identites des voisins, ou connaissance seulement de l'orientation dans un reseau complet, etc);

- selon que le reseau lui-même est synchrone ou asynchrone,etc.

Dans ce chapitre, est présenté un echantillon d'algorithmes publiés pour les differentes topologies de reseau: anneau, a maillage complet, quelconque. Parmi ces algorithmes certains ont été évalués sous Echidna.

1 ALGORITHHES D'ELECTION POUR TOPOLOGIE ANNEAU

1.1 Introduction

La plupart des algorithmes étudiés ne s'interessent qu'au cas où les processus du reseau sont dotes de numeros d'identification uniques. De plus, ils supposent tous que le reseau des processus ainsi que le systeme de communication sont fiables. Parmi ces algorithmes, il y en a ceux qui sont basés sur une structure en anneau unidirectionnel asynchrone [CHAN 79, DOL 82, PET 821, en anneau unidirectionnel synchrone [FRE 87b, GAF 891 avec une complexité en O(n) messages, ou bidirectionnel asynchrone [HIR 80, LAV 89, BOD 86, FRA 82, LEE 871.

Deux algorithmes sont donnés dans [GOL 871 avec l'hypothese que le reseau bidirectionnel de connexion des processus pourrait être defaillant. Dans le premier algorithme le nombre n de procesus n'est pas connu, cependant, chaque processus Pi connait les identites IDi-1 et IDi+1 de ses deux voisins de gauche et de droite. Le second algorithme suppose la connaissance de n. La complexité attendue pour ces deux algorithmes est O(nlogn) messages. En effet, cette borne est la meilleure borne pour tous les algorithmes proposes dans le cas d'un reseau asynchrone(c'est la complexité minimale possible dans le cas d'un tel reseau [PAC 84]).

Parmi les alqorithmes présentés dans ce chapitre, quatre algorithmes sont évalués sous Echidna. L'algorithme de Chang et Roberts [CHA 791, c'est l'un des premiers algorithmes proposes pour resoudre le problème d'élection. L'algorithme d'Hirschberg et Sinclair [HIR 801, base sur un anneau bidirectionnel. Le troisieme algorithme est celui propose, simultanement, par Dolev [DOL 821 et Peterson [PET 821 sur un anneau unidirectionnel necessitant O(nlogn) messages. Ceux-ci présentent une refutation de la conjecture de Hirschberg et Sinclair qui avaient énoncé que seul un anneau bidirectipnnel pouvait permettre la realisation d'une compléxité inferieure a $O(n^2)$ messages. L'algorithme de KORACH [KOR 811, par contre, est une variante probabiliste de l'algorithme de [CHA 791 conçue pour un anneau bidirectionnel.

1.2 Hypothèses

Les hypotheses faites sur le réseau de processus et le systeme de communication:

- Les processus ont des fonctionnements asynchrones et ne communiquent que par messages.

- Ils ont des identites differentes prises dans un ensemble totalement ordonné.

- Aucun processus ne tombe en panne pendant l'election.

- Aucun message n'est perdu, ni altéré.

- Les délais de transmission sont quelconques, mais finis.

- Enfin, les processus n'ont pas d'informations sur le nombre n de processus, et ne connaissent que leurs identites.

Dans le cas oh une ou plusieurs de ces hypotheses ne sont pas respectées, cela sera precise.

Tous les processus exécutent le même texte, sauf que chaque texte fait reference a l'identité du processus qui l'exécute. Ces identites sont toutes distinctes. Les processus ont des comportements pouvant être distincts, l'algorithme a donc une symétrie de texte.

Remarque:

Les algorithmes sont présentés en commençant d'abord par les algorithmes basées sur une structure unidirectionnelle puis bidirectionnelle et en tenant compte de l'ordre chronologique de leur publication.

1.3 Algorithme De Chanq Et Roberts [CHA 791

1.3.1 Principe

L'algorithme est base sur l'hypothèse que les processus sont structures en anneau unidirectionnel de façon quelconque. Chaque processus Pi, qui connait son propre numéro i, le transmet a son voisin de gauche Pj. Celui-ci, a la reception d'un tel message compare son propre numéro j au numero reçu i. Si ce dernier est plus grand, il le transmet a son voisin de gauche, dans le cas contraire c'est son propre numéro qui est transmis: c'est le principe de l'extinction sélective de messages.

Initialement un ou plusieurs processus commencent une election en envoyant un message: un tel processus se marque comme participant a l'election. L'arrivee d'un message a un processus non marqué, provoque sa participation à l'election. Le processus marqué, qui reçoit un message portant son propre

numero conclut alors que son numero a fait un tour complet sur l'anneau et est donc le plus grand; il est alors élu et diffuse son identite aux autres si cela est necessaire au fonctionnement de l'algorithme qui a déclenché l'election.

1.3.2 Description formelle

a) Contexte local d'un processus Pi

Constante	mon-numero: valant i;
Variable	participant: bouleen initialise a faux;
	coordinateur: entier; (utile si l'algorithme qui a declenche l'election nécessite la connaissance du vainqueur final)

b) Types de messages

- Election: indique que la valeur qu'il transporte est candidat a l'election.

- Elu: necessaire pour diffuser l'identité du gagnant.

c) Algorithme exécuté par un processus Pi

```
Lors de
   decision de provoquer une election
     faire   participant:= vrai;
             env-vg(élection.mon-numéro)
     fait;
   reception de (élection.i)
     faire selon le cas
             i>mon-numéro:  env-vg(electi0n.i);
                            participant:= vrai;
             itmon-numero   et non-participant:
                            env-vg(élection.mon-numéro);
                            participant:= vrai;
             i=mon-numero:  env-vg(élu.i);
     fait;
   reception de (élu,i)
     faire  coordinateur:- i;
            participant:= faux;
     fait;  si i<>mon-numéro alors env-vg(C1u.i) fsi;
```

. complexité thdorique

En messages: l'algorithme de Chang et Roberts utilise, pour 1'étape d'élection, un seul type de message. Dans le cas favorable (les processus sont places sur l'anneau dans 1 'ordre croissant de leurs numéros, et le processus Pmax lance l'election) le nombre de messages est 0(n). Dans le cas le plus defavorable (ordre decroissant et tous les processus lançent l'election) n(n-1)/2 soit $0(n^2)$ messages sont necessaires. Dans les autres cas le nombre moyen est de 0(nlgn).

En temps: le temps requis pour l'election est de O(n).

1.4 Alqorithme De Dolev, Klawe Et Rodeh [DOL 821

Ce même algorithme fût publié simultanement par Dolev [DOL 821 et Peterson [PET 821.

1.4.1 Principe

- Les communications se font sur un anneau unidirectionnel de droite a gauche,
- les messages sont recus dans leur ordre d'emission.

L'algorithme procède par phases, a chaque phase l'anneau unidirectionnel des processus est reduit au moins de moitie (un processus sur deux devient passif) jusqu'a ce qu'il ne reste qu'un processus actif qui aurait determine alors l'identite du processus élu.

Le principe d'une phase est le suivant:

Un processus Pj attend d'avoir les numéros associes à see deux voisins de droite, soient Pk et Pl. Puis il se place du point de vue de Pk (qui est entre Pj et Pl) il teste si celui-ci. est maximum local auquel cas il transmettra le numero, sinon Pj deviendra passif et son role se limitera alors a transmettre les messages reçus (lorsque Pj reste actif son voisin de droite devient necessairement passifl.

1.4.2 Description formelle

a) Contexte local d'un processus

Constante mon-numero valant j;
Variable Ctat: (actif, passif) initialisé a passif;
max : entier initialise a mon-numero; {numéro du processus du point de vue duquel le processus Pj se place}
voisindroit: entier; {identité du premier voisin actif a droite de Pj}

b) Types de messages

Deux types de messages sont utilises pour véhiculer les identites des deux processus voisins de droite actifs.

(b, numéro) où b vaut 1 ou 2 selon qu'il s'aqisse du premier ou du deuxieme voisin de droite.

c) L'algorithme exécuté par Pj

```
Lors de
  decision de provoquer une election
    faire   env-vg(1,mon-numéro);
            état:= actif;
    fait;

  reception de (1, numero)
    faire si état=actif alors
                si numéro<>max alors
                    env-vg(2,numéro);
                    voisindroit:=  numero;
                sinon (max est le plus grand numero
                    il est diffuse aux processus)
                fsi
          sinon env-vg(1, numero);
        fsi;
    fait;

  reception de (2, numero)
    faire si état=actif alors
              si voisindroit>numéro et voisindroit>max
                  alors max:= voisindroit;
                        env-vg (1, voisindroit)
                  sinon état:= passif;
            fsi
                  sinon env-vg(2, numero);
        fsi;
    fait;
```

. **Complexité théorique**

A chaque phase de l'algorithme, l'anneau est reduit de moitie et il se termine quand il ne restera qu'un seul processus actif dans l'anneau. Ce qui nécessite (Logn+1) phases. 2n messages sont @changes (2 par processus (1, numéro) et (2, numéro)) durant les Log n phases, et n messages dans la derniere phase fun seul processus est actif et il n'y a donc pas un deuxieme voisin). La complexité de l'algorithme est, par consequent, de 2nLogn + n messages.

1.5 Algorithme De Hirshberg Et Sinclair [HIR 801

1.5.1 Principe

Les processus sont places de façon quelconque sur un anneau bidirectionnel.

Le principe de l'algorithme est celui d'election primaire sur des ensembles de processus de plus en plus grands jusqu'à contenir tous les processus places sur l'anneau. Ainsi, lorsque Pj lance une election, il se declare candidat et cherche à savoir si son numero est supérieur a ceux de ses deux voisins immédiats Pk et Pl (dont il ne connait pas les numéros) en leur diffusant son propre numero. Si l'un des deux autres processus Pk ou Pl est vainqueur (k ou 1 supdrieur a j), il devient candidat s'il ne l'est pas deja. Si les deux autres processus se déclarent battus, Pi réitère sa candidature sur un ensemble plus grand. Les messages de candidature vont progresser a chaque iteration sur des chemins dont la longueur croit selon une puissance de 2.

Le role d'un processus battu a une iteration donnée se limite a faire suivre les messages d'un côté vers l'autre sur . l'anneau.

1.5.2 Description formelle

a) Contexte local d'un processus Pi

Constante	mon-numero valant i;
Type	position=(non-concerné, candidat, battu, élu);
Variable	état: position initialisé à mon-numero;
	lgmax: entier;
	vainqueur: entier;
	nbrep: 0.. 2 initialise a 0;
	repok: booléen initialisd à vrai;

b) Types de messages

Deux types de messages sont utilises: candidature et réponse.

1) (candidature, numero, lg, lgmax);
- numero indique le numero associe au processus qui a lance le message de candidature,
- lgmax indique la longueur du chemin que le message doit à priori parcourir sur l'anneau,
- lg indique la longueur deja parcourue.

2) (réponse, bool, numero);
- bool a la valeur vraie si la réponse est favorable, faux sinon,
- numero désigne l'identite du processus déstinataire (c'est une façon de l'adresser sur l'anneau puisque ce numero est unique).

c) L'algorithme exécuté par un orocessus Pi

```
Lors de
  decision de provoquer une election
    faire etat:= candidat; lgmax:= 1;
    tant que etat=candidat
      faire nbrep:= 0; repok:= vrai;
      envoidouble(candidature, mon-numero, 0, lgmax);
      attendre nbrep=2;
      si non repok alors etat:= battu fsi;
      lgmax:= lqmax * 2;
    fait;
  fait;
  reception de (réponse, bool, numero)
    faire si numéro=mon-numéro alors nbrep:= nbrep+l;
                                    repok:= repok et bool;
    sinon fairesuivre(réponse, bool, numéro);
    fait;fsi;
  reception de (candidature, numero, lg, lgmax)
    faire selon le cas
      numerotmon-numero:
                répondre(réponse, faux, numéro);
                si état=non-concerné alors
                fsi;          provoquer election
      numéro>mon-numéro:
              état:= battu; lg:= lg+1;
              si lgtlgmax alors
              fairesuivre(condidature,numéro,lg,lgmax)
                sinon répondre(réponse, vrai, numero)
              fsi;
      numéro=mon-numéro:
              si état<>élu alors
                    état:=élu;
                    vainqueur:= mon-numero;
    fait;     fsi;  fairesuivre(terminé, numero);
  reception de (terminé, numero)
    faire si vainqueur<>numéro alors
                fairesuivre(terminé, numéro);
                vainqueur:= numero;
    fait; fsi;  état:= non-concern&;
```

Remarque sur l'adressage:

La technique d'adressage implicite des processus places sur 1 'anneau est assurée en utilisant la longueur du chemin. Pour un message de réponse, le destinataire, qui a une identité unique, se reconnait par son numero place dans le message.

. Complexité theorique

Au debut, bien que(dans le pire des cas) n processus soient candidats sur des chemins de longueur 1, on a au plus $n/(2^{i-1}+1)$ processus qui seront candidats sur des chemins de longueur 2^i.

Or un prqcessus lançant sa candidature sur des chemins de longueur 2^k provoque le transfert de quatre messages (2 dans chaque direction). Par consequent, le nombre maximum de messages échangés est borne par 8n+8nLogn ce qui est O(n Log nl.

1.6 Alqorithme-P De Korach, Rotem Et Santoro [KOR 81]

1.6.1 Principe

L'anneau est bidirectionnel, l'ordre des messages est conserve et les delais de transmission des messages sont tous égaux.

L'algorithme execute par un processus Pi procède en étapes. Une variable locale Max garde l'identitb maximale reçue par Pi. Les étapes d'election se déroulent comme suit: A la réception d'un message contenant l'identité IDj du processus Pj venant de la droite ou de la gauche de Pi: si IDj est supérieur a Max, le message est retransmis et continue son parcours. Si deux messages arrivent simultanement des deux directions, celui qui véhicule la valeur la plus petite est ignoré. Le processus Pi qui reçoit une valeur IDi égale a Max est elu. Il diffuse un message de fin d'élection.
L'initialisation consiste a choisir, avec une probabilité eqale a 1/2, la direction droite ou gauche vers laquelle le message contenant IDi est émis par le processus Pi.

1.6.2 Description formelle

a) Contexte d'un processus Pi..

Constante IDi: entier valant i;
Variable max: entier initialisé a i;{identité de l'élu}
participant, elu: bouleens initialisés a faux.

b) Types de messages

Deux types de messages sont utilisés: (numéro) pour la phases election, et (stop) pour signaler fin election.

c) L'algorithme exécuté var un processus Pi

```
Lors de
   decision de provoquer une election
     faire Max:= IDi; élu:= faux; participant:= vrai;
     choisir une dierction d appartenant a {droite, gauche};
     I emettre(IDi) sur d;
     fait;

   reception de (numéro)
     faire Cas participant: si numéro> Max alors
                              max: = numero;
                              fairesuivre(numero)
                            sinon si numero=max alors
                                    !     env-vg(stop)
                                  fsi;
                            fsi ;

                 non-participant:  provoquer election avant de I
                       traiter le message;
          Fcas;
     fait;

   reception de (stop)
     faire
      I si élu=vrai alors fairesuivre(stop) fsi;
     fait;
```

. **Complexité théorique**

Le nombre moyen de messages prévu par les auteurs [KOR 811 est 3n Hn/4 (Hn=0.693..Logn) et n /2 dans le pire des cas, mais dans [Lav 89C] la valeur asympotique calculée est $\sqrt{2}$ nHn/2+O(n). Cette evaluation a été faite en supposant un fonctionnement synchrone des processus.

1.7 Alqorithme De Fredreickson Et Lynch Pour Un Réseau Synchrone [FRE 87b]

. **Principe**

L'algorithme se deroule sur un anneau unidirectionnel avec communication synchronisee par des tops horloge.

Il est initialise par des processus qui décident independamment les uns des autres de se réveiller. . Le réveil ne se fait pas necessairement en même temps pour tous les processus.
Une fois actif, un processus (appelé maintenant participant) génère un message, qui se déplacera sur l'anneau, transportant son identité IDi.

L'algorithme est base sur deux idees. La **première-idée** consiste a transmettre les messages a des vitesses differentes. Le message comportant l'identité IDi circule a la vitesse d'un transfert tous les 2^i tops c'est-a-dire que le message est retenu pour une durée de 2^{i-1} tops au niveau de chaque processus, qu'il visite, avant d'être retransmis.

Un message qui est rattrapé par un autre message plus rapide est éliminé de la course. De même, lorsqu'un message transportant un IDi, atteint un processus j, il est éliminé si j<i et si, en plus j aurait deja généré son propre message. L'idée de faire transferer les messages a des vitesses variables est Cgalement exploitee dans [GAF 85a] qui propose un algorithme semblable a celui-ci.

La **deuxième-idée** est d'avoir une phase preliminaire avant la phase des vitesses variables. Dans cette phase, tous les messages circulent a la même vitesse: un transfert par top. Quand un processus decide de participer a l'election, il envoie un message vers son voisin. Ce dernier est transmis sur l'anneau jusqu'à la rencontre du participant suivant. Des cet instant, le message entre en deuxieme phase et l'algorithme se poursuit comme expliqué précédemment.

. **Complexité théorique**

En messages elle est de $O(4n)$. En temps elle est de $O(n2^i)$ rounds, oh i est l'identité du processus Clu et c'est la plus petite identite du reseau.

1.8 Algorithme De Spirakis, Tampakas Et Athanasios [SPI 891

Cet algorithme suppose connue au moins une borne superieure sur la valeur de n. Dans [SPI 891 est considéré le cas d'un anneau de processus dont les identites ne sont pas toutes distinctes: le reseau est anonyme.

. **Principe**

L'algorithme est base sur un anneau unidirectionnel et consiste en deux Ptapes majeures:

(a) Etape de creation de candidats: dans cette, etape les processus, choisissent par tirage aléatoire une identité dans l'intervalle [1,..,n]. A la fin de cette phase, seul un ensemble de processus deviennent candidats. Les autres deviennent des relais dont le role se limitera, par la suite, a retransmettre les messages reçus. Dans cette étape, il existe une probabilite de blocage d e l'algorithme lorsqu'aucun processus ne devient candidat.

(b) Etape d'election: Les candidats echangent des messages comportsnt un numero de phase, et leur identite ID. Au debut de chaque phase, apres le choix de ID, et incrementation du numero de la phase, un message contenant ces informations est emis. Si

le message reussit a parcourir tout le reseau sans rencontrer un autre candidat dont la phase et/ou l'identite sont superieures aux siennes, alors son initiateur devient élu. Sinon une autre ID est choisi et le message est réémis après incrementation de la phase.
Un processus candidat i recevant un message (phasej, IDj) superieur à (phasei, IDi) n'est plus candidat. IL réemet le message (phasej, IDj).

. **Complexité théorique**

Le trafic engendré est en moyenne O(n) messages.

1.9 Algorithme De Franklin [FEW 821

. Principe

La communication est synchrone. Cet algorithme est base sur un principe analogue à celui de Dolev [DOL 821, mais ici l'anneau est bidirectionnel. Une etape de l'algorithme se deroule comme suit:

Un processus actif teste lui-même si son numero est plus' grand que ceux de ses deux voisins de gauche et de droite. Si ce n'est pas le cas, il devient passif.
A chaque etape, l'anneau des processus actif voit sa taille divisée au moins par deux: les processus passifs relayant simplement les messages reçus d'un côté vers l'autre.
L'algorithme se termine lorsqu'un processus reçoit un message de lui-m&me.

. **Complexité théorique**

Comme à chaque etape, il y a 2n messages transmis et il y a (Logn+1) étapes, au total 2n(Logn+1) messages sont échangés plus n messages pour informer l'ensemble des processus de l'identite du vainqueur.
Si l'unité de temps d'éxécution est supposee être le delai de transfert d'un message d'un processus vers ses voisins, alors dans le pire des cas, le temps d'éxécution est 0(nLogn).

1.10 Algorithme-D De Bodlaender Et Van Leeuwen [BOD 861

. Principe

C'est une version déterministe de l'algorithme-P décrit ci-dessus. L'idée est de remplacer la phase d'initialisation qui choisie une direction de façon aleatoire, 'en une phase completement déterministe. Chaque processus Pi envoie le message contenant IDi dans la direction oh se trouve le voisin de plus petite identité(au debut de l'algorithme, un processus Pi émet un message contenant IDi vers ces deux voisins et reçoit a son tour IDi-1 et IDi+1). Ceci va éliminer les plus petits voisins. Dès la fin de la phase d'initialisation, seul les processus qui sont maximums locaux survivront.

. **Complexité théorique**

$\sqrt{2}$ nHn/2 messages en moyenne et $n^2/4$ dans le pire des cas.

1.12 Alqorithme De Leeuwen Et Tan [LEE 871

. **Principe**

A chaque direction sur l'anneau est associee une file d'attente de messages et l'ordre des messages est preserve.

L'algorithme opère de la maniere suivante:
A tout moment chaque processus tient a jour un registre ID contenant la plus grande identite de processus, et un registre DIR booléen qui pointe vers la "direction" de l'anneau oh se trouve un processus de plus petite identite. Le "grand" candidat se trouvera sur un côté du processus tandis que DIR pointera dans la direction opposée. Les messages contenant le "grand" candidat sont envoyés dans la direction oh se situe le "petit" candidat encore actif. L'idee est donc de se debarasser du candidat dont l'identite est la plus petite. C'est la même idée qui est utilisee dans l'algorithme-D dans la phase d'initialisation.

. **Complexité théorique**

La complexite en messages est de 1,44...nLogn + 0(n).

2 ALGORITHHES POUR TOPOLOGIE COHPLETE

2.1 Introduction

Dans un reseau complet chaque noeud est connecté a tous les autres noeuds (n-l liaisons sont reliées a chacun des noeuds). Ce type de reseau est modélisé par un graphe G=(V, E) complet.

L'algorithme d'election le plus evident dans ce cas est le suivant: Chaque noeud envoie des messages contenant son identite a tous ses voisins. Ces derniers choisissent comme élu celui dont l'identite est la plus élevée. La complexite de cet algorithme est $0(n^2)$ messages dans le pire des cas.

Pour les algorithmes présentés dans cette section, la complexite en messages est améliorée, Cependant, la complexite temporelle augmente vu que la fréquence de transmission des messages est réduite.

Les algorithmes rencontres, conçus pour la topologie complete, sont bases sur des reseaux a communication bidirectionnelle. Comme pour le cas de l'anneau, rares sont ceux qui resistent a certains types de pannes [GAR 821. Les algorithmes proposes dans [AFE 85] présentent une complexite en 0(nLogn) messages et la meilleure borne temporelle en 0(n) unites de temps. L'idee de base exploitee dans ces algorithmes

[AFE 85, KOR 84] est la notion de quantité-de-travail qui est utilisee pour choisir le candidat a éliminer parmi les processus concurrents a l'election. L'algorithme de Loui [LOU 861 par contre utilise la notion du sens d'orientation pour reduire la complexite à O(n) messages. Cette notion a été reprise dans [ATT 89, MAS 891 pour, respectivement, des reseaux en anneau auxquels des liaisons supplementaires ont été rajoutées, et des reseaux oh peuvent apparaitre des pannes de processeurs.

2.2 Hypotheses

Les hypotheses suivantes sont communes a tous les algorithmes, d' autres hypotheses seront précisées en cas de besoin.

- A chaque noeud est associe un numero identificateur unique (ID) puisé dans un ensemble totalement ordonne.

- Au niveau de chaque processus, chaque port d'entree et chaque port de sortie a un identificateur unique connu par le processus.

- Initialement (avant le debut de l'algorithmel, en dehors des identificateurs de ses ports, un processus n'a aucune information sur le réseau. En particulier, il ignore les identificateurs des processus auxquels il est connecté. Les processus n'ont pas de connaissance quant à la topologie du réseau ou au nombre total de processus.

- Les liaisons de communication sont fiables. Les messages émis sur ces 1 ignes atteignent leur destination en un délai imprevisible, mais fini.

- L'ordre des messages est preserve.

- Les liaisons sont bidirectionnelles.

- Le mode de communication est asynchrone.

- Les algorithmes présentés sont caracterises par une symetrie de texte.

Les algorithmes seront présentés, comme pour la topologie en anneau, dans l'ordre chronologique de leur publication.

2.3 Algorithmes De Garcia-Molina [GAR 821

L'etude faite dans [GAR 821 sur la définition.de l'election suivant le type de pannes peut être considérée comme une premiere classification de ce type d'algorithmes. Dans cette etude, l' auteur a défini l'election suivant deux environements d'execution, un environement avec des pannes de communication (formation de sous réseaux) et des arrêts temporaires de sites et un environement exempt de ces deux types de pannes. 11 propose un algorithme pour chaque type d'environement,

respectivement, l'algorithme "brutal" et l'algorithme "invitation".

2.3.1 Principe de l'algorithme brutal

Dans le premier algorithme, dit "brutal"(bully algorithm). Les hypotheses suivantes sont considérées:
- Le reseau de communication est fiable.
- Les messages sont traités par les processus dès leur arrivée avec une durée nulle.
- Une hypothèse implicite de connaissance globale est egalement faite, les processus connaissent les identités des autres processus ainsi que la structure du reseau.

L'algorithme consiste, alors pour un process Pi, a envoyer un message de candidature à tous les processus Pj tels que i<j<=n. 11 se declare battu et il attend une candidature pendant un temps T au delà duquel il relancera sa candidature. Si par contre au bout de la premiere période T, il ne reçoit aucune réponse, il conclut que ces processus sont en panne. 11 demande aux processus Pk tels que k<i d'être a l'écoute car il ne connait pas leur état et il les informe ensuite de son election en leur envoyant un message Clu.

Le nombre maximum de messages générés par cet algorithme est en $0(n^2)$.

2.3.2 Principe de l'algorithme(invitation)

Dans le deuxième algorithme, il est suppose que le réseau peut se partitionner en plusieurs sous-réseaux et qu'un site peut arrêter son traitement et le reprendre ensuite comme si rien ne s'est passe. On peut donc avoir a un instant donné plusieurs groupes de processus indépendants avec chacun un coordinateur. Le principe de l'algorithme consiste alors pour un coordinateur d'un groupe donné à inviter dès que possible(dès la r&apparition d'un site isolé et/ou le retablissement d'une ligne en panne) les autres coordinateurs a rejoindre son groupe. A la reception d'une invitation, un processus peut accepter ou refuser une telle invitation. Au terme de l'algorithme, si le reseau est redevenu connexe, un seul coordinateur subsistera.

2.4 Algorithme De Korach, Moran Et Zaks [KOR 84]

. Principe

Initialement tous les processus se considerent rois. Au terme de l'algorithme, seul le processus élu restera dans cet état, tous les autres processus deviendront "citoyens". Durant l'execution de l'algorithme, chaque roi est racine d'un arbre representant son royaume. Les autres noeuds de cet arbre sont citoyens du royaume. Un roi tente d'agrandir son royaume en envoyant des messages vers d'autres rois, leur demandant de joindre leurs royaumes au sien.
Six types de messages sont utilisés:

réveil: un noeud qui reçoit ce message se reveille et demarre l'algorithme. Au plus un message réveil atteint un processus quelconque.
demande: ce message est envoyé par un roi i, a travers un canal non utilise, au roi j pour l'inviter a rejoindre son royaume.
acceptation: ce message est émis par le roi i, comme reponse au roi j pour l'informer de l'acceptation de son invitation.
mis-a-jour: un roi i reçevant un message d'acceptation emet le message mis-a-jour, pour aviser ses nouveaux citoyens.
citoyen: ce message est émis par un citoyen en reponse a une demande de son roi.

. **Complexité théorique**

L'algorithme gènère au maximum 5nLogk+0(n) messages (k étant le nombre de processus ayant reçu le message réveil).

2.5 Algorithmes de Afek et Gafni [AFE 85]

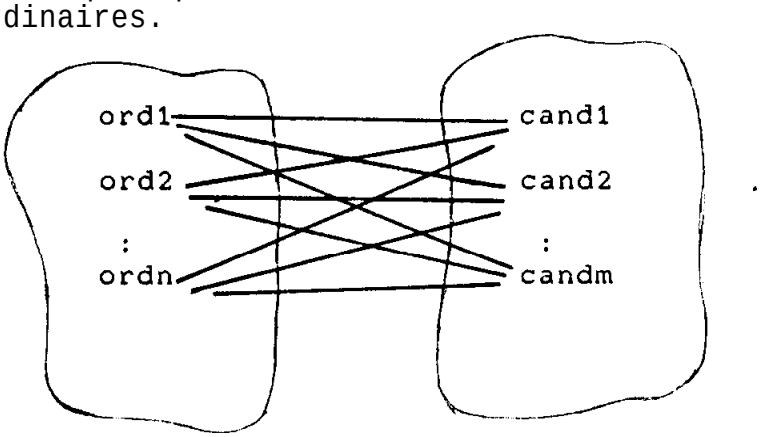
Trois algorithmes asynchrones A, B et C ont été proposes dans [AFE 851. Chacun d'eux est conçu dans le but de resoudre les problemes poses par le precedent, de telle façon que l'algorithme C atteigne la complèxité la plus interessente par rapport à B et a A. Les trois algorithmes sont tous bases sur le même principe.

2.5.1 Principe

L'algorithme est initialise par un ensemble quelconque de noeuds candidats à l'élection. Chaque candidat tente de construire une arborescence, dont il est la racine. Les autres noeuds seront tous ses fils(directs car la topologie est complete). Pour cela il envoie des messages sur toutes les lignes qui lui sont connectées. Un seul candidat sera élu, tous les autres seront elimines.

Un noeud se reveillant spontanement génère deux processus:
- un processus candidat,
- un processus ordinaire.

Tous les autres noeuds génèrent seulement un processus ordinaire. Chaque processus candidat est relié à tous les processus ordinaires.



Les candidats utilisent une variable appelee 'niveau' qui reflete le nombre de noeuds battus. Ces derniers tiennent a jour des variables 'niveau', qui gardent la trace du candidat ayant le niveau le plus élevé, qui ils ont été visites.

Un candidat qui atteint un noeud dont le niveau est plus élevé que le sien est éliminé. Au contraire, si le niveau du candidat est plus élevé ou égal a celui du noeud hôte, le niveau de ce dernier est remplacé par le niveau du candidat qui devient alors son père(racine), et qui tente d'eliminer l'ancien père.

Les trois algorithmes different en deux points:
(1) la façon dont la variable niveau est mise à jour,
(2) la regle d'élimination.

Dans l'algorithme A, le niveau est égal a u nombre de processus ordinaires battus. Dans l'algorithme B, il est égal au nombre de candidats concurrents éliminés, e t dans l'algorithme C, il represente la combinaison des deux precedents.

Deux variables de type pointeur, pere et pere-potentiel, sont tenues a jour par chaque processus ordinaire. Elles . representent, respectivement, le père le plus recent et le candidat qui, actuellement, tente de joindre le processus en question a son arborescence afin de devenir son nouveau père.

2.5.2 Algorithme A

Dans cet algorithme, la variable **niveau** d'un candidat e s t le nombre **de processus ordinaires dont il est la racine.** Elle est increment&e de un, lorsque le candidat a reussi a agrandi *r* son arborescence d'un noeud. La regle utilisee est la suivante:

Pour que P puisse capturer un noeud V dont le proprietaire courant est le candidat Q:

```
(1) (niveau, ID) de P doit être plus grand (lexicographiqu-
ement) q u e (niveau,ID) du père courant de V.
(2) l'ancien père de V doit être éliminé.

Autrement dit:
Si (niveau(P), ID(P)) < (niveau(V), ID(Q)), P e s t éliminé.
S i (niveau(P),ID(P)) > (niveau(V), ID(Q)) a l o r s
                        (1) V reçoit niveau de P.
                        (2) P est envoyé vers Q.
Arrivé a Q:
S i (niveau(P),ID(P)) < (niveau(Q), ID(Q)), P est éliminé.
Si Q a été éliminé par un autre candidat alors capturer V.
S i (niveau(P), ID(P)) > (niveau(Q), ID(Q)) a l o r s
                        (1) Q est éliminé.
                        (2) P capture V.
```

. Description formelle

a) Contexte-local d'un processus-Pi

Constante ID: entier valant i; (* identite du noeud i *)

Variable niveau: entier initialise a 0;
ID-proprietaire: entier initailisé à 0 ;
non-traverse: ensemble des liaisons adjacentes;
père, pere-potentiel: pointeurs initialises à nil.

b) Types de messages

Un seul type de messages est utilisé: [niveau, ID).

cl L'algorithme-exécuté-par-un-noeudi

```
Candidat:
 Lors de
 déçision de provoquer une election
 faire
 l l:= elt(non-traversé); (* un element dans non-traverse *) I
 I envoyer(niveau,ID) à 1 ; (* 1: lien de communication *)
 fait;
 reception de (niveau', ID') depuio 1'
  faire si (ID=ID') alors (* capture réussie *)
  |    |    niveau:= niveau+l;
  |    |    non-traverse:- non-traverse-l;
  |    |    si non-traverse <> ϕ alors
  |    |                    l:= elt(non-traversé);
  |    |                    envoyer(niveau,ID) sur 1 ;
  |    |       sinon annoncer"élu, f i n  algorithme";stop
  |    |    fsi
  |    | sinon si ((niveau',ID')<(niveau,ID)) alors
  |    |              ignorer ce message;
  |    |          sinon (1) envoyer(niveau',ID') sur 1 ' ;
  |    |                (2) ignorer tous les messages
  fait; fsi;    fsi ;        antérieurs
```

```
I Ordinairc:
I Lors de reception de (ID',niveau') sur 1'
  Faire cas (niveau',ID')<(niveau,ID-propriétaire): i g n o r e r
  |  |    |                          le message;
I I    |   (niveau',ID')>(niveau, ID-proprietaire):
I I    |       père-potentiel<---ID';
I I    |       niveau<--- niveau';
I |    |       si père=id then père<--- pere-potentiel fsi;
|  |    |       envoyer (ID', niveau') vers père;
I I    |    (niveau',ID')=(niveau, ID-propriétaire):
|  |    |       père<--- pere-potentiel;
  Fait; fais;   evoyer(ID',niveau') v e r s père;
```

. **Complexité théorique**

Le nombre de messages est (4nlgn=2,773nLogn), la complexité temporelle est O(n).

2.5.3 Algorithme B

La variable niveau représente ici le nombre total de processus candidats éliminés, c'est-à-dire, le nombre de racines battus.

Pour qu'un candidat p puisse capturer le noeud v, il faut que la condition niveau(v) > niveau(p) soit vérifiée.
Quand le processus candidat p a atteint le processus ordinaire v qui est déjà capturé par un autre candidat q:

```
| Si niveau(p) < niveau(v), p est éliminé;                          |
| Si niveau(p) > niveau(v), v est capture par p,                    |
|              niveau(v):= niveau(p);                               |
| I xi niveau(p) = niveau(v), p est envoy6 à q.                     |
|   Arrivé à q :                                                    |
|    S i (niveau(p),ID(p)) < (niveau(q), ID(q)), p set 61iminb.I    |
|    Si q a déjà été éliminé, p est éliminé aussi car celaI         |
|      signifie qu'il exite un candidat x dont la variable|         |
|      niveau est supérieure en valeur à celle de p.                |
|    S i (niveau(p),ID(p)) > (niveau(q), ID(q)), a l o r s          |
|        (1) q est éliminé,                                         |
|        (21 p incrémente sa variable niveau,                       |
|        (3) p capture v.                                           |
```

. Description formelle

a) **Contexte-local-d'un-vrocessus-Pi**

Constante ID: entier valant i; (* identité d u noeud*)
Variable niveau: entier initialisé à 0 ;
ID-propriétaire, id-potentiel: entiers initailisés à 0 ;
non-traverse: ensemble des liaisons adjacentes;
père, père-potentiel: pointeurs initialisés à nil.

b) **Types-de-messages**

Un seul type de messages est utilisé: (niveau, ID).

c) L'algorithme exécuté par un noeud i

```
Candidat:
 Lors de
 déçision de provoquer une élection
  faire
    e:= elt(non-traverse);  { un element dans non-traverse }
    envoyer(niveau,ID) sur e;
  fait;
 réception de (niveau', ID') sur e'
  faire si (ID=ID') alors { capture reussie }
                    niveau:= niveau+1;
                    non-traverse:=  non-traverse-e;
                    si non-traverse 0 ϕ alors
                                e:= elt(non-traversé);
                                envoyer(niveau,ID) sur e;
                      sinon annoncer"élu, fin  algorithme";stop
                    fsi
        sinon si ((niveau',ID')< (niveau,ID)) alors
                     ignorer ce message;
                     attendre un autre message;
                   sinon
                     (1) envoyer(niveau',ID') sur e';
                     (2) ignorer tous les messages futures;
  fait; fsi;     fsi;
```

```
Ordinairc:
 niveau<--- -1;
 Lors de la reception de (niveau', ID') sur e'
  faire cas niveau'< niveau : écarter message;
           niveau'> niveau : { remplacer le père }
                 père:= e';
                 niveau<--- niveau';
                 ID-propriétaire:= ID';
                 id-potentiel:= 0;
                 père-potentiel:= nil;
                 envoyer(niveau',ID') sur e';{ vers le père } I
           niveau'= niveau:
                si (ID'< ID-proprietaire) alors
                  père:= père-potentiel;
                  niveau': = niveau'+1;
                  ID-proprietaire:= ID';
                père-potentiel:= nil;
                  envoyer(niveau',ID') vers le père;
                I sinon
                  si pere-potentiel <> nil
                    alors écarter le message
                    sinon { pas de père potentiel }
                      id-potentiel:= ID';
                      père-potentiel:= e';
                      envoyer (niveau', ID') vers le père;
 Fait; fcas; fsi; fsi;
```

. **Complexité théorique**

2nLogn+2n messages et O(nLogn) unites de temps sont attendus.

La croissance de la complexite temporelle est due au fait que, a la difference de l'algorithme A, le niveau d'un candidat n'est pas une fonction du nombre de noeuds qu'il a deja captures. Un candidat qui realise un "travail" important(et consomme plus de temp.31 en construisant une arborescence, pourrai t être éliminé par un candidat ayant réalisé un travail moins important. L'aborescence qui aurait été construite par ce dernier serai t moins volumineuse, mais le nombre de racines qu'il aurait battues serait plus élevé.
L'algorithme A quant à lui, est caracterise par le fait qu'un noeud puisse être éliminé plusieurs fois(voir régle d'extinction de l'algorithme A).
Dans l'algorithme C ces deux problèmes sont résolus par la combinaison des deux techniques.

Le but fixé par l'auteur consite a ameliorer la complexite temporelle de l'algorithme B, en conservant la complexite en messages.

2.5.4 Algorithme C

Une fois qu'il a battu un processus ordinaire un candidat incrémente s a variable niveau qui sera au moins égale a Log(nombre total de noeuds battus). Plus precisement, la variable niveau est mise à jour comme suit:

niveau: = maximum (Log(nombre de noeuds non racine battus (processus ordinaires)), nombre de noeuds racines battus).

Cette regle d'utilisation de la variable niveau permet d e prendre en compte le travail, effectué par un candidat en nombre de processus ordinaires et/ou candidats concurrents battus.

. **Descrition** formelle

La description formelle de l'algorithme C est similaire a celle de B. Une variable taille est utilisee pour compter le nombre de noeuds captures par candidat. Le seul changement qui est introdui t c'est la mise à jour de la variable niveau d'un processus candidat (première clause then):

```
taille := taille+1;
niveau:= Max(niveau', Log (taille));.
```

. **Complexité théorique**

Les compléxités ont été estimees a 2n(Logn+Log Logn+2) messages, et O(nLog Logn) unites de temps.

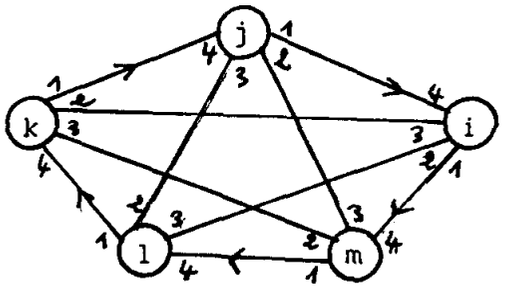
Remarque:

Le temps peut être amélioré à O(n) unites de temps en apportant la modification suivante à l'algorithme [AFE 851: les candidats qui atteiqnent un niveau plus élevé que Logn tenteront de capturer n/Logn noeuds simultanement (en une unite de tempsl.

2.6 Algorithme de Loui, Matsushita et West [LOU 861

2.6.1 Hypothèse suplémentaire

on admet que les processus ont un sens d'orientation globale.
Definition: On se fixe un cycle hamiltonien comprenant tous les processus. Chaque liaison est étiquetée par la distance le long de ce cycle, au processus se trouvant a l'autre extrémité d e 1 'arc. En particulier, si le processus x est a la distance d du processus y, alors y est a la distance (n-d) de x.



2.6.2 Principe

Initialement tous les processus sont actifs, ils deviendront tous passifs sauf l'élu (qui a l'identite la plus élevée). Un processus actif devient passif a la reception d'un message provenant d'un processus dont l'identité est supérieure a la sienne. Un processus actif qui reçoit un message émis par un processus dont l'identité est inférieure a sa propre identité ID, lui envoie en reponse un message contenant cette ID. Ceci permet d e s'assurer que le processus en question passera dans l'état passif.

2.6.3 Description formelle

a) Contexte-local-d'un-processus-Pi

Constante ID: entier valant i; {identité du processus Pi }

Variable D: entier initialisé a n-1;
E , Newid: entiers;
état: (actif, passif, élu) initialise a actif;

b) Les brimitives dc communication

Envoyer(d:e,j): émet le message (e,j) lc long de la liaison d au processus qui se trouve a la distance d.

Recevoir(e,j): attend l'arrivee du message (c,j).
c: distance de l'emetteur au recepteur,
j: identité de l'emetteur.

c) Algorithme execute oar un orocessus Pi

```
Lors dc la
  decision de provoquer une election
    faire
    I envoyer(n-D, n-D, ID);  (* D a été initialisé à n-l *)
    fait;

  reception dc (c, j)
    faire
    I Newid:- j;
    I si état=actif alors
    I I                D:= c ;
    I I                cas Newid < I D : envoyer(n-D, n - D , ID);
    I |                    Newid = I D  : état:= élu;
    | |                                   annoncer "élu" ;
    I I                    Newid > ID : état:= passif;
    I I                fcas;
    I I              sinon { état=passif }
    I |                 E:= e;
    | I                 envoyer(n-D,(n-D+E), Newid)
    I fsi;
    fait;
```

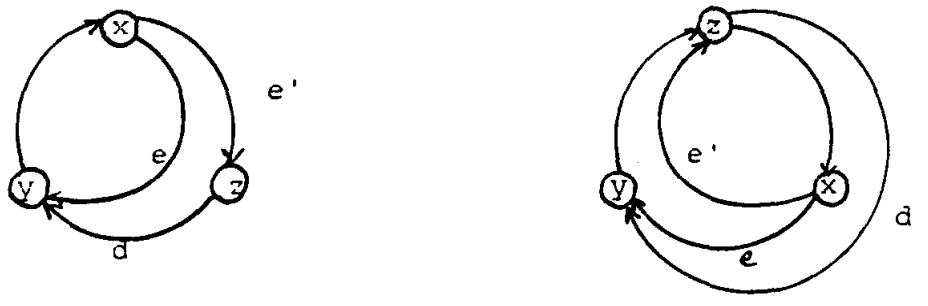
. Compléxité theorique

Le nombre de messages est inferieur a 3,62n(c-à-d C(n)) ct le temps est borne par $n+\log_{\phi} n+O(1)$ unites de temps, où $\phi=\frac{1}{2}(1+\sqrt{5})$.

2.6.4 L'algorithme corrige

L'algorithme donné par les auteurs (voir description detaillee ci-dessus) n'est pas correct(voir annexe). L'erreur se situe au niveau du calcul de distance. En effet, l'idée sur laquelle est base l'algorithme consiste en ce qui suit:
Dans l'etat passif, un processus y recevant un message (e,j) provenant du processus x(donc y est à la distance e de x), envoie un message à destination du processus z, qui l'avait, a un moment anterieur, mis dans l'etat passif. Le processus z se trouve à la distance N-d de y, et a la distance e' de x. Il faut noter que le message envoyé par le processus passif y, evitera aux processus actifs de lui envoyer d'autres messages par la suite. Apres l'émission de ce message, aucun processus

n'emettra un message à y. La distance e' incluse dans le message envoyé par y à z, permettra a z de communiquer directement avec x sans transiter par y(qui est passifl.
En tenant compte du sens d'orientation, deux schéma(figure 111.11 sont possibles, selon que c'est x ou z qui precede immediatement y le long du cycle.



(a) e'=e-d (b) e'=N-d+e

Figure III.1

Or dans l'algorithme, la valeur prise en compte est celle où la situation (b) se presente.
L'algorithme n'aboutit pas si le cas(a)se présente. Le même alqorithme est reproduit avec la modification de la clause:

```
sinon {état=passif}
  E:=e;
  envoyer(n-D, (n-D+E), Newid)
```

qui devient:

```
sinon { état=passif }
  E:= e;
  si (e-d)>0 then envoyer(n-D,(e-D), Newid)
             else envoyer(n-D,(n-D+E), Newidl
  fsi;
```

Remarque :

Masuzana et al. ont propose dans [MAS 891 une version modifiée de l'algorithme de [LOU 86]. Cet algorithme a été transformé de façon à resister aux pannes de processus. L'idee sur laquelle est basée cette transformation consiste, pour un processus donné, a utiliser (fp+1) jetons au lieu d'un jeton. Le nombre fp(<n/2) reprdsente les processus défectueux. En effet, l'utilisation de fp+1 jetons, permet de pallier au problème d'interblocage, car les jetons envoyésà des processs deffectueux sont perdus.

2.7 Algorithme De Attiya, Leeuwen Et Shmuhel [ATT 891

Les auteurs de [ATT 891 exploitent ici la notion de sens d'orientation pour généraliser le travail présenté dans [LOU 861. Ils presentent un algorithme general d'élection pour des reseaux dont la topologie est l'anneau mais auquel, des liaisons ont été ajoutees. Les cas extremes étant l'anneau et le reseau a topologie complete (n-1 cordes ont été ajoutees a l'anneau [LOU 86]). Ils proposent un corollaire qui montre que l'ajout de 0(logn) cordes a chaque processus du réseau suffit pour obtenir un algorithme utilisant au plus 0(n) messages.

. Principe

L'idée utilisé est la même que celle de [LOU 861, et le principe de base est celui de [FRA 821. A chaque phase, un ensemble de processus est candidat à l'élection. Durant l'étape i un processus qui est maximum local(son ID est supérieur a ceux de ses deux voisins actifs les plus proches), passe à l'etape i+1 et échange son ID avec ses deux voisins actifs. Ces derniers sont repérés sur l'anneau grace à un échange de messages permettant a chaque processus qui survit a une étape donnée de mesurer la distance, le long de l'anneau, entre lui et ses deux voisins actifs les plus proches.

3 ALGORITHHES D'ELECTION POUR TOPOLOGIE QUELCONQUE

3.1 Introduction

Dans cette partie, sont présentés des algorithmes s'executant sur des reseaux modélisés par un graphe quelconque. D'une part ces algorithmes calculent un extremum, et d'autre part, ils construisent une arborescence couvrante sur le graphe permettant a tous les processus de communiquer via des chemins uniques avec le processus élu associé a la racine.

Pour la topologie quelconque, les algorithmes conçus peuvent être divises en deux grandes classes: des algorithmes proposes pour des reseaux anonymes, et des algorithmes conçus pour des reseaux dont les processus sont dotes d'identites toutes distinctes. Dans la deuxieme catégorie, il y a les deux algorithmes proposes par [HEL 86b], bases respectivement sur les techniques d'exploration de reseaux en profondeur et en parallele. Ils utilisent la notion de transfert de connaissances avec une complexité théorique en $0(n^2)$ dans le pire des cas.
S. Skyum [SKY 871 a Cgalement publié un algorithme base sur une technique d'exploration de reseaux en profondeur. Ce dernier atteint une complexité en 0(nLogn)+0(E) messages et c'est aussi la complexité suggérée dans [LAV 89I].

Dans le cas de reseaux anonymes, on peut citer les alqorithmes de Matias et Afek [MAT 891 dont la complexité est en 0(ErLogn Logrl messages oh r=1/E et (1-E) est la probabilite

avec laquelle un leader unique est choisi. De son côté Lavault [LAV 871 avait propose un algorithme qui construit un arbre couvrant le réseau, la racine étant le processus élu. Sa complexité est de $n^2+O(n)$. Un autre algorithme est propose par [LAV 90]. 11 choisit un Clu avec une probabilité de 1-E et une coniplexite de O(E+nLogn).

3.2 Hypotheses

- Le graphe G(V,E) est quelconque et les liaisons sont bidirectionnelles.
- Il n'y a pas de perte de messages.
- Les messages sont délivrés au bout d'un temps non previsible mais fini, de plus, ils ne sont pas altérés.
- La communication est asynchrone.
- Au cours du calcul distribué, un processus quelconque n'apprendra jamais la structure globale du reseau, ni même le nombre total de processus participants: sa connaissance topologique se limitera toujours à ses voisins directs.
- Il y a symétrie du texte de l'algorithme execute par l'ensemble des processus.

Ces hypotheses sont en general respectées, dans le cas contraire, d'autres hypotheses seront donnees.

3.3 LES SITES SONT DOTES D'IDENTITES DISTINCTES

Les alqorithmes proposes dans ce paragraphe, supposent que chaque processus est dôté d'une identite unique.

3.3.1 Algorfthme de Helary, Haddi Et Raynal [HEL 86b] (exploration en profondeur)

3.3.1.1 Principe

En plus des hypotheses citées precedemment, cet algorithme admet le desequencement de messages. Le principe sur lequel repose l'algorithme propose consiste pour un processus Pi, a lancer une exploration du reseau dans le but de visiter tous les processus en etablissant des chemins de ceux-ci vers lui-même. Comme l'algorithme peut être démarré par un ou plusieurs processus, une ou plusieurs explorations peuvent être simultanement en cours. A chacune de ces explorations est associe un poids correspondant au processus initiateur de celle-ci.

Une exploration Ek (issue de Pk) peut être vue comme une activité sequentielle; une sequence de messages est engendree, chaque message est traité par le processus qui le reçoit et celui-ci, a son tour, peut Cventuellement emettre un nouveau message de poids k. A chaque instant de l'exploration, un seul message de poids k est en transit sur une ligne ou en traitement par un processus. Les auteurs decomposent le processus d'election en étapes matérialisées par la reception d'un message m par un processu Pi:

1. Etape d'extinction:

-ou bien Pi a deja été atteint par une exploration de poids superieur a k (ou a déjà lance sa propre exploration Ei si i>k),

-ou bien Pi n'a pas encore été atteint, mais i>k.

Dans un cas comme dans l'autre, l'exploration Ek est Cteinte par Pi; de plus, dans le deuxieme cas, Pi lance l'exploration de poids i.

2. Etape de conclusion:

Dans cette étape et les deux suivantes, on suppose que Ek est l'exploration de poids le plus fort ayant jamais atteint Pi; max designera la plus grande identite de l'ensemble des processus du réseau. L'information de contrôle amenee par le message indique a Pi que tous les autres processus ont été atteints par l'exploration. Dans ce cas, Ek s'arrête et on a alors nécessairement: k-max; Pi peut donc conclure que l'exploration Emax a établi une arborescence recouvrant le reseau et les autres processus sont informés de la fin de l'algorithme par un signal diffuse sur les branches de cette arborescence.

3. Etape de generation:

Pi doit faire progresser Ek, et possede des voisins non encore atteints par Ek. Il met alors a jour les liaisons relatives à Ek, et engendre un nouveau message d'exploration de poids k vers un de ses voisins non atteints (par exemple le plus grand).

4. Etape de renvoi:

Pi doit faire progresser Ek; tous ses voisins ont été atteints par Ek, mais l'information de contrôle amenee par le message indique que des processus du rdseau n'ont pas encore été atteints par Ek. Pi engendre alors un message de poids k permettant de remonter vers un processus non encore atteint.

3.3.1.2 Description formelle

al **Contexte** local d'un processus **Pi**

Constante voisins: initialisé à{identités des voisins de Pi};

Variable état: (initial, candidat, battu, élu) initialisé à initial;
pqvu: entier initialisé à i;
{donne l'identité de la plus grande exploration, a la fin pgvu=identité maximum)

pred : entier initialise a nil;
succ: ensemble d'entiers initialise a A;

b) **Type de messages**

(i) explorer(k,z,s): **k** represente l'identite du processus ayant initialise l'exploration, **z** contient les identites des processus **visités** par ce message, et **s définit** l'ensemble des processus voisins **immédiats** des processus de z.

(ii) rebrousser(k,z,s): les cahmps de message sont les même que ceux de type explorer.

(iii) conclure servent a signaler la terminai son de l'algorithme.

c) **L'algorithme exécuté** oar **un** processus Pi.

```
Procedure lancer-exploration
 debut soit x=maximum(voisins);
     état:=candidat;
       pred:=nil;
       succ:={x};
       envoyer explorer(i,{i},voisins-{i}) a Px
 fin;

Lors de
  decision de provoquer une election
    faire
    I si état=initail alors lancer-exploration fsi
    fait;
  reception de explorer(k,z,s) depuis Pj
  faire
    cas pgvu>k: si état=initial alors
                    lancer-exploration fsi;
     pgvu<k: étati:= battu;
             pgvu:= k ;
             pred:=j;
             soit y=voisin-z;
           cas y=Φ : succ:=Φ ;
             cas s=Φ : envoyer conclure à Pj;
                  s<>Φ: envoyer rebrousser(k,zU{i},s)
             fcas;          à Pj;
                y<>Φ: soit x=maximum(y);
                      succ:= {x};
  fait;fcas;     fcas;      envoyer explorer(k,zU{x}) a Px
```

```
reception de rebrousser(k,z,s) depuis Pj
   faire si pgvu-K alors
      |    soit y-voisins s;                                  I
      |    cas y=Φ : envoyer rebrousser(k,z,s) a Ppred
      |         y<>Φ: soit x-maximum(y);
      |               succ:=succU{x};
      |               envoyer explorer(k,z,s-{x}) a Px
      |    fcas;
      |  fsi;
   fait;

 reception de conclure depui Pj
   faire si pgvu=i alors état:= Clu fsi;
      |   ∀ x∈ (succ U pred)-{j}: envoyer conclure a Px
   fait;
```

. **Complexité théorique**

Dans le cas le plus favorable, oh Pmax est seul a lancer l'exploration:

. n-l messages du type explorer,

. 0 à n-l messages du type rebrousser (0 est obtenu dans le' cas d'un reseau en anneau],

. n-l messages du type conclure.

soit 3(n-1) messages(2(n-1) pour un reseau en anneau).

Dans le cas le plus defavorable:

. au plus $\left(\sum_{k=1}^{n-4} k'\right)+(n-1)$ messages du type explorer,

. autant messages rebrousser,

. et, n-l messages conclure,

soit (n-1)(n-3) messages tous types confondus. La complexite est donc $O(n^2)$ dans le pire des cas.

La complexite temporelle est de [2(n-1)+d]D, où d est le diametre de l'arborescence construite (1<=d<=n-1), et 2(n-1)D le temps de transfert des messages explorer et conclure. dD est le temps maximum pour réveiller Pmax. D est le délai d'acheminement des messages entre deux processus quelconques.

Pour un reseau complet, le nombre de messages est 3(n-1), et (2n-3)D le temps d'election.

Pour un reseau en anneau, le nombre de messages est n/2(n/2+1)-1 c'est-a-dire O(n), et le temps (3n-4)D.

3.3.2 Algorithme de Helary, Maddi Et Raynal [HEL 86b] (exploration en largeurl

3.3.2.1 Principe

Dans cet algorithme l'exploration issues d'un processus Pk est lancée en parallèlle vers tous ses voisins. Comme precedemment un processus ne laisse progresser vers ses voisins la composante de l'exploration qui l'atteint que si le poids k de celle-ci est supérieur au poids de la derniere exploration prise en compte.

Pour obtenir une arborescence couvrante, il est nécessaire que tout processus autre que Pmax ait un seul predecesseur et que deux processus distincts aient des ensembles de successeurs disjoints.

Tout processus Pi, lorsqu'il reçoit un message relatif a une exploration qu'il prend en compte pour la premiere fois, mémorise l'emetteur Pj du message comme Ctant son predecesseur. Il propage l'exploration vers ses voisins, et attend d'avoir reçu des acquittements de tous ceux-ci, avant d'emettre l'acquittement vers pj. Dans le cas où il reçoit un message relatif a l'exploration deja prise en compte, il l'acquitte immediatement en indiquant a Pj de ne pas l'inclure dans ses successeurs.

3.3.2.2 Description formelle

a) <u>Contexte local d'un processus Pi</u>

La constante voisins et les variables @tat, pgvu, pred, succ ont la même signification que dans le cas de l'exploration en profondeur.

variable nbacq: entier non negatif initialise a 0;
{permet de memoriser le nombre d'acquittements attendus et de contrôlre l'envoi de l'acquittement vers son *prédécesseur*}

b) <u>Type de messages</u>

(i) explorer(k,z): k represente l'identité du processus ayant initialisé l'exploration, z désigne l'ensemble des processus dont on est sûr qu'ils ont été ou seront visités par des mesages relatifs a cette exploration}

(ii) acquitter(k,x): x permet au recepteur de savoir s'il doit considérer Pi comme un de ses successeurs dans l'arborescence(x=terminé) ou non (x=déjàvu).

c) L'algorithme exécuté par un processus Pi

```
procedure lancer-exploration
 début
 |      &tat:-candidat;
 |      pred:=nil;
 |      succ:=voisins;
 |      nbacq:=cardinal(succ);
 |      ∀ x∈ succ: envoyer explorer(i,voisins-{i}) à Px
 fin;

lors de
 décision de provoquer une élection
    faire
    | si état=initail alors lancer-exploration fsi
    fait;
  réception de explorer(k,z) depuis Pj
    faire
    | cas pgvu>k: si état=initial alors
    | |                  lancer-exploration  fsi;
    I I                                                          I,
    | |   pgvu-k : envoyer acquitter(z,déjàvu) à Pj;
    | I   pgvu<k: état:= battu;
    | |           pgvu:= k;
    | |           pred:= j;
    | |           succ:= voisin-z;
    | |            cas succ=∅: envoyer acquitter(z, terminé)
    I I                                  à Pj;
    I I                 succ<>∅: nbacq: = cardinal(succ);
    | |                 ∀ x∈ succ: envoyer acquitter(k,zUsucc)
    | |                                  à Pj;
    | fcas;         fcas;
    fait;
  reception de acquitter(k,x) depuis Pj
    faire  si pgvu=K alors
    |      |   si x=déjàvu alors succ:=succ-{j} fsi;
    |      |   nbacq:=nbacq-1;
    |      |   si nbacq-0 alors
    |      |   | cas k-i:  état:= élu;
    |      |   | |     k<>i: envoyer acquitter(x,terminé) à
    |      |   | |                   Ppred
    |      |   | fcas;
    |      fsi;fsi;
    fait;
```

. Complexité théorique

La complexité dans le pire des cas est en O(n) messages, et de 3dmaxD en temps. dmax est la distance maximale de Pmax à tout autre processus (1<=dmax<=n-1). 2dmaxD est le temps de transfert des messages explorer et acquitter, relatifs à l'exploration Pmax, et dmaxD le temps nécessaire pour réveiller Pmax.

3.3.3 Alqorithme de Sven Skyum [SKY 871

. Principe

S. Skyum propose un algorithme base sur une technique de parcours de réseaux en profondeur. Quand un processus v se reveille, il crèe un jeton ‹s,0,v› de niveau 0 et ce jeton commence un parcours d'exploration en profondeur. Le symbole s indique si le jeton circule en avant, en arriere ou chasse un autre jeton. Chaque fois qu'un jeton fait un pas en arriere sur un arc, soit il fermera l'arc, soit il le declarera appartenant a l'arbre couvrant et il ne sera plus jamais traverse par un message.

Si un jeton ‹s,l,u› émis par un processus u atteint un processus qui, a deja été traverse par un jeton d'un niveau superieur, le jeton ‹s,l,u› est eteint. S'il arrive a un processus qui a été visité par le jeton ‹s,l,w› de même niveau, alors:

- ou bien u est inferieur a w, le jeton ‹s,l,u› s'arrete en v et attend qu'un autre jeton le rattrappe. Ceci est réalisé en. mettant le processus v a l'état (candidat, niveau l).

- ou bien u est superieur à w, le jeton ‹s,l,u› arretera son parcours et poursuivra ‹s,l,w› aussi longtemps qu'il n'est pas chasse, a son tour, par un troisieme jeton de niveau 1, auquel cas ‹s,l,v› s'arrêtera et l'état du processus oh a lieu l'arrêt devient (candidat, niveau 11.

Lorsque deux jetons de même niveau se rencontrent, un nouveau jeton de niveau l+1 est créé, et un nouveau parcours en profondeur du graphe est lance, alors que les deux jetons de niveau 1 sont eteints. Si un jeton reussit a terminer son parcours, il declarera le processus initiateur comme Clu et on a un arbre de recouvrement qui relie tous les processus au processus élu.

. Complexité théorique

Elle est de 2E+2nLogn+0(n) messages, où E représente le nombre de liaisons connectant les procesus du réseau.

3.2.4 Algorithme de Lavallee et Lavault [LAV 8911

. Principe

Cet algorithme construit un arbre couvrant et designe la racine de l'arbre comme l'élu. L a construction de l'arbre couvrant se fait, progressivement, par combinaison de sous-arbres en fragments de plus en plus grands, en respectant la relation d'ordre total strict existant dans l'univers I auquel appartiennent les n identités.

Au début, chaque processus constitue un fragment dont il est la racine. Le processus de connexion de fragments consiste en ce qui suit:
a) un fragment F donné est candidat à la fusion avec un autre fragment G, et la racine de F Cmet une requête de connexion vers G.

b) une telle requête est accéptée(ou rejetée) par le premier processus du fragment G qui l'aurait reçue, en tenant compte de la relation d'ordre strict existant entre les identites.

. **Complexité théorique**

Cet algorithme possède, dans les reseaux à maillage complet, une complexite en messages de 1/2nLogn+O(n) dans le pire des cas et une complexite de O(n) en moyenne. Pour la topologie en anneau, dans le pire des cas, la complexite en messages est 3nLogn+O(n), et en moyenne, elle est égale à 1/2nLogn+O(n).
Dans les reseaux d'interconnexion quelconque, un bon majorant de la complexite en messages est 2E+7/2nLogn+O(n), mais dans le cas oh n est connu, la borne superieure est de 7/2nLogn+0(n).
Pour le temps la compléxité est de O(Log n) unites.

3.4 ALGORITHME POUR RESEAUX ANONYHES

Sous l'hypothèse d'identités distinctes des sites du réseau, le probleme d'election a été ramene au problème de calcul d'un extremum et diverses solutions ont été proposees. Si l'on ne s'astreint pas à l'hypothese d'unicité des identites des sites (soit à cause de l'absence d'identités, soit parce que les identités existent mais ne sont pas toutes distinctesl, le probleme d'election d'un site ne peut être résolu par une méthode de calcul d'extremum.

Dans un réseau anonyme, initialement tous les processus n'ont aucune identité et, chaque processus ne connait que les ports le reliant à ses voisins directs. Au cours de l'algorithme, il n'apprend d'eux que leurs identites. Il est montré dans [ANG 80] que si les sites sont tous anonymes (ou possèdent tous une identité commune), il n'existe alors aucune solution déterministe au probleme de l'élection distribuée, que le reseau soit synchrone ou asynchrone. Par conséquent, toute solution au probleme ne saurait être obtenue, dans ce cas, que par un algorithme non déterministe. Santoro et Lavallée [SAN 86, LAV 90] soulignent que le problème de l'election peut Ptre résolu par un alqorithme probabiliste si les hypotheses de symetrie de sites et de leur connaissance de n (nombre total de sites) sont faites conjointement.

3.4.1 Algorithme de Christian Lavault [LAV 871

Les processus connaissent un majorant du nombre de sites, Maxint, fixé par la machine.

. **principe**

La premiere etape de l'algorithme est l'etape d'activation-denomination. En effet, initialement tous les processus sont dans un même état dit état initial. Aprbs le passage spontane des processus vers l'etat actif, comme ils sont anonymes, leur premiere tâche consiste à se donner une identite par generation d'un nombre aleatoire dans [1..Maxint]. Donc, plusieurs processus peuvent acquerir des identites egales. Après le choix de l'identite, les processus passent dans l'état candidat.

Dans la deuxieme etape est construite une arborescence couvrante du graphe G associe au réseau. Un ou plusieurs fragments peuvent Ptre simultanement construits sur G. Ces fragments seront fusionnes deux à deux, chacun cherchant à en absorber un autre, jusqu'à obtention d'une arborescence constituée de tous les processus du réseau.

Les identites des processus sont, comparées selon la relation d'ordre total non strict existant entre elles du fait de l'existance d'identités non-distinctes.

Dans la mesure oh il est possible que plusieurs processus (et, eventuellement tous les processus, avec une probabilite infime de 1/(maxint) puissent tirer aleatoirement le même numéro d'identite entre 1 et Maxint, le processus de négociation-connexion peut échouer et la terminaison correcte de l'algorithme peut en être affectée. Pour Cviter cela, une procedure permettant de réaliser autant de nouveaux tirages aléatoires d'identités qu'il est nécéssaire a été prévue.

Par exemple, un sommet quelconque Pi (racine ou non), reçoit une demande de négociation de la part d'une racine quelconque dans le réseau possedant la même identité, l'algorithme suivant est execute:

1. le sommet Pi renvoie un message de refus a la racine qui demande la négociation.

2. Tant qu'il reste encore un fils sortant Px de Pi possédant une arête sortante non-utilisee et d'identité différente de celle de Px(puisque Px est fils de Pi ==> IDx=IDi), la négociation est tentée a travers Px.

   Par contre, lorsque chaque fils sortant Px de Pi ne possede que des portes sortantes dont les identites sont égales a l'identite de Px, alors Pi doit changer d'identité et effectuer un nouveau tirage aleatoire uniforme dans

[1,..Maxint]-{IDi}.

3. L'cnscmblc fils-sortants finit par ctrc vidc, et il ne reste PIUS de portes a explorer. Il reste alors une seule racinc qui diffuse un message fin a tous les sommets.

. **Complexité théorique**

Le nombre de messages attendu est minoré par 11n/2+O(n); il s'agit du cas où toutes les identites sont toutes distinctes. Il est majoré par n +O(n); c'est le cas où toutes les identites sont egales.
Le temps d'execution est minor.6 par $O(n\sqrt{11n}Logn)$, et majore par $O(Logn\sqrt{11n})$.

Remarquesurlaterminaison:

11 y a une autre façon d'assurer la terminaison de l'algorithme lorsque chaque noeud connait la taille n du système distribué. Comme chaque racine r connait la taille de son propre fragment F, alors que F est fusionne a un autre fragment. G; il envoie sa taille a la racine de G. Ainsi, si un fragment atteint eventuellement la taille n, sa racine se declare Clue, et l'algorithme est terminé. Une variable compteur de noeuds est initialisee a la valeur 1 et mise a jour après chaque fusion entre deux fragments.

3.4.2 Algorithme de Yossi Matias et Yehuda Afek [MAT 891

. **principe**

Le processus Pi reçevant une identite IDi diffuse a tous ses voisins toutes les identites qu'il a reçues auparavant, sont inferieures a IDj. Cela signifie que c'est le processus portant la plus grande identité qui capture le réseau. De plus, chaque processus mémorise la liaison par laquelle il a reçu le plus grand ID, ce sera son processus-père. 11 y aura donc la construction d'une forêt couvrante.

Si le plus grand ID est unique, alors la forêt couvrante se réduit a un arbre dont la racine est le leader.

La difficulté de l'algorithme reside dans la procedure dc selection d'un ID. Elle est conçue de façon a ce que deux conditions soient satisfaites simultanement avec une grande probabilité:
(a) un processus unique est distingué parmi les autres noeuds;
(b) le nombre d'identites distinctes est faiblc.

La procedure choisir(IDi):

Chaque processus Pi sélectionne deux nombres ti et si de la façon suivante:
(1) ti: est le nombre de lances avant l'obtention de face;

(2) si: est un nombre choisi au hasard dans l'intervalle [1..d], oh d=(O(rlogr) avec r=1/ε et 1-ε étant la probabilite de succès de l'algorithme.

Donc l'identité du processus Pi est IDi=‹ti, si›.

L'algorithme ELECT exécuté par Pi

```
| Procedure initialise(i)                                          |
| debut                                                            |
|   ! appel choisir(IDi);                                          |
|   ! max:=IDi; père:=0;                                           |
|   ! diffuser IDi vers tous les voisins.                          |
|  fin.                                                            |
|  Lors de la reception de ID a travers la liaison 1               |
|  faire                                                           |
| !  si IDi n'est pas défini alors appel initialise(i) fsi;        |
| !  si ID <= max  alors  ignore                                   |
| !  !  !              sinon  max:-id;  père:=l;                   |
| !  !  !                     diffuser ID a tous ses voisinn       |
| !  fsi;                                                          |
|  fait.                                                           |
```

. **Complexité théorique**

L'algorithme ELECT s'arretera Cventuellement. Apt-es la terminaison! **un** leader unique est choisi, et **un** arbre **couvrant** est construit avec la probabilite (1-ε). L a complexite en messages attendue est O(Erlogn.logr), E est égal **au nombre** *d'arcs* du graphe des processus. La complexité temporelle est O(D) oh D est le diametre du réseau. Ces complexités sont donnees avec une probabilite superieure a (1-ε).

3.4.2.3 **L'élection avec** detection de la terminaison

L'algorithme ELECT a été modifié de façon a permettre la detection de la terminaison. En effet, une hypothese supplémentaire(H) a été introduite:

H: **la valeur** n de la taille du réseau est supposee etre bornée inferieurement par L, et superieurement par 2L: (L< n <=2L).

Après la terminaison de I'alqorithme ELECT, Max a la même valeur au niveau de **tous** les noeuds, et les liaisons parentes des noeuds constituent une **forêt couvrante** du reseau avec un ou plusieurs arbres. L'algorithme a reussi l'élection d'un leader si la **forêt obtenue** *ne* **contient** qu'un seul arbre.

Donc si L< n <=2L, **l'abre** dont la racine est le noeud élu doit avoir une taille superieure a L. De plus, s'il existe un arbre dont la taille est superieure à L dans la forêt, on est certain qu'il est le **seul** a avoir cette taille, par consequent, l'élu est unique.

3) Degré de répartition(ou symètrie)

Tous les algorithmes étudiés possedent la propriété de symetrie de texte au sens défini dans [RAY 851.

4) Connaissance de l'état global ou local

E1. Les processus connaissent la taille n du reseau ou une borne sur n.
E2. Les processus ne connaissent ni la taille n du reseau ni une borne sur n.
E3. Les processus connaissent la topologie du reseau.
E4. Les processus ne connaissent rien sur la topologie du réseau.
E5. Les processus connaissent les identites de leurs voisins immediats.
E6. Les processus connaissent les identf tés des ports d'entrée/sortie auxquels ils sont connectés.

5) Topologie en anneau

Design.1 algo.	Hypo. communic.	Hypo. réseau	pannes	info. glob./loc.	complexités messages	temps
[CHA 7 8 1	C2, C4, C5 C6, C9	S1	C13, S3	E2, E4, E6	n(n-1)/2 IO(n) IO(nlgn)	O(n)
[DOL 8 2 1	C2, c4, c5 C6, C 7, C9	S1	C13, S3	E2, E4, E6	2nLogn+n pire	/
[HIR 801	C1, C4, C5 C6, C9	S1	C13, S3	E2, E4, E6	O(nLogn) pire	/
[KOR 8 1 1	C1,C4,C5,C6 C7, C9, C12	S1	c13, S3	E2, E 4, E6	I $n^2/2$ nLogn	/
[BOD 861	mêmes que ci-dessus	S1	c13, S3	E2, E4, E 6	$\sqrt{\frac{3}{2}}n^2/4$ nLogn	/
[SPI 8 9 1	C2, c4, C5 C6, C9	S2	c13, S3	E1, E 4, E5	O(n) moy.	/
[FRA 8 2 1	C1, c3, C5 C6, C9	S1	C 1 3, S3	E2, E 4, E6	2nLogn+3n pire	/
[LEE 8 7 1	C1, c 4, C5 C6, C 7, C9	S1	IC 1 3, S 3	E2, E 4, E6	4,44Logn+O(n) . pire	/
[GOL 8 7 1	C1, c4, C5 C6, C l0	S1	C14, S3	E1, E4, E6	O(nLogn) pire	/
[FRE 87b]	C2, c3, C5 C6, C9	S1	C13, S3	E2, E4, E6	O(4n) pi re	$O(n^2)$

6) Topologie complete

Design. algo.	Hypo. communic.	Hypo. réseau	pannes	info. glob./loc.	complexites messages	temps
[GAR 82] Brutal	C1, C4, C5, C6, C7, C9, C11	S1	C13, S4	E1, E3, E5	$O(n^2)$ pire	/
[GAR 82] Invit.	C1, C4, C5, C6, C7, C10	S1	C14, S4	E1	$O(n^2)$ pire	/
[KOR 84]	C1, C4, C5, C6, C7, C13	S1	C13, S3	E2, E4, E6	5nLogk+On k=nb initl	/
[AFE 85] Algo. A	C1, C4, C5, C6, C7, C9, C13	S1	C13, S3	E2, E4, E6	2,773nLogn	O(n)
[AFE 85]B	même	même	même	même	2nLogn+2n	/
[AFE 85]C	même	même	même	même	2nLogn+LgLogn+2	/
[LOU 861	même + orient.	S1	C13, S3	E2, E4, E6	3.62n	
[ATT 891	même que [LOU 861	/	/	/	O(n)	/

71 Topologie quelconque

Design. algo.	Hypo. communic.	Hypo. réseau	pannes	info. glob./loc.	complexites messages	temps
[HEL 86b] Profond.	C1, C4, C6, C8, C9	S1	C13, S3	E1, E4, E5	3(n-1) (n-1)(n-3)	O(n)
[HEL 86b] Largeur	même quel ci-dessus	même	même	même	$O(n^2)$	/
[SKY 87] Invit.	C1, C5, C6, C9	S1	C13, S4	E2, E4, E6	$2E+2nLogn+O(n)$	
[LAV 89I]	C1, C4, C5, C6, C7, C9	S1	C13, S3	E2, E4, E5	nLogk+On 3nLogn+On	
[LAV 871	C1, C4, C6, C7, C9	S2	C13, S3	E1, E2, E4, E6	n+O(n) n^2+O(n)	
[MAT 891	C1, C4, C6, C9	S2	C13, S3	E2, E4, E6	O(D)	(+)

(+): D délai d'acheminement des messages entre deux processus quelconques.

VII Les techniques utilisées

Dans ce paragraphe sont resumes les techniques et les mecanismes utilises dans la conception des alqorithmes d'election. Le problème d'election peut se ramener à deux sous-problèmes:

1. construction d'un algorithme de parcours,

2. construction d'un algorithme d'tlection utilisant l'algorithme defini en (1.) afin de transmettre les messages vers tous les processus impliqués dans I'election.

1 Election quand les processus sont dotés d'identités distinctes

Dans le cas d'un réseau en anneau, le parcours consiste, pour un processus initiateur à transmettre un message vers un de ses deux voisins. Ce message est transmis le long de l'anneau jusqu'à atteindre son initiateur. Les algorithmes d'election qui ont été étudiés sont bases sur le principe d'extinction selective.

Extinction sélective

Chang et Roberts [CHA 791 ont propose, en 1979, le principe de l'extinction selective des messages pour un anneau unidirectionnel.
L'algorithme de Chang et Roberts comprend les trois phases suivantes:

1. La recherche du processus initiateur de plus fort numero et l'arrêt des autres candidatures par extinction selective. Pour cela chaque initiateur diffuse un message de reconnaissance contenant son numero i. Si un tel message arrive sur un processus initiateur j de plus fort numero, il n'est pas transmis, ce qui annule la candidature de i. Si i>j, le message est propagé par j. Il en est de même si le processus j n'est pas initiateur.

2. La fin de l'election: elle est obtenue quand le message d'un site initiateur i a parcouru tout l'anneau et est revenu à son initiateur: i=j. Seul le message du processus de numéro maximal peut faire le tour de l'anneau sans être annul&.

3. L'annonce de l'election: le processus élu fait circuler un message de fin de reconnaissance, avec son numéro, sur tout l'anneau; il informe ainsi tous les processus, de son identité et de la fin de l'election.

Franklin, Hirshberg et Sinclair [HIR 80, FRA 821 ont adopté le principe de l'extinction selective au cas de l'anneau bidirectionnel.

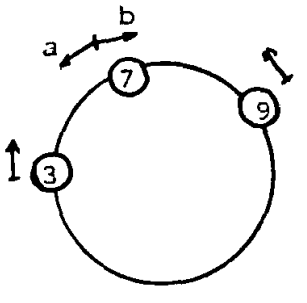
De nombreux auteurs ont propose des ameliorations à ces alqorithmes initiaux et ont abouti parfois a des alqorithmes très complexes [DOL 82, PET 82, GOL 87, ROT 87, LEE 87].

La spécificité des methodes d'extinction selective les unes par rapport aux autres repose [KAI 881 dans la phase 1. 11 faut Climiner au plus vite les petits numeros. Dans tous les cas, l'unicite de l'élu résulte de l'unicite de la numirotation des processus, de l'unicite du processus dont le message de reconnaissance fait le tour de l'anneau et de l'unicite du message de fin d'election. Les performances varient d'une election a une autre puisqu'elles dependent de la numérotation relative des processus.

Pour éliminer au plus vite les petits numeros il est possible d'appliquer une des methodes suivantes:

Méthode 1: Dans la variante probabiliste de l'alqorithme de Chanq et Roberts proposée dans [KOR 811, l'idée qui a été utilisee sur un anneau bidirectionnel consistait a modifier la phase d'initialisation en choisissant une direction d € {droite, gauche) avec la probabilite 1/2. Le message contenant l'identite du processus sera émis dans la direction d.

La compléxité dans le pire des cas est, pratiquement, la même que celle de l'alqorithme unidirectionnel, mais un message sera éliminé plus rapidement s'il se diriqe vers un message se déplaçant dans la direction opposee. La figure suivante [MAT 881 illustre une telle situation:

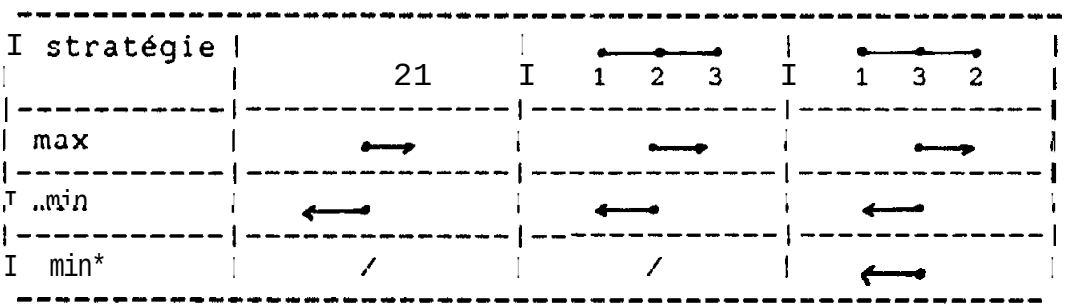


Si dans l'anneau unidirectionnel le message émis par le processus 3 est éliminé par le 7, alors sur l'anneau bidirectionnel il est possible de s'attendre a ce que les deux messages se rencontrent a mi-chemin si le 7 decide d'envoyer son message dans la direction opposée (cas al. Donc, avec la probabilite 1/2 le message 3 se déplace seulement la moitié de la distance, ce qui reduit le nombre total de messages moyen. Cependant, si 7 envoie son message dans la même direction que 3(cas b), il existe une probabilite superieure a zéro tel qu'un autre processus(9 dans la figure ci-dessusl rencontre le message émis par 3 avant d'atteindre le processus 7.

Méthode 2: 11 est possible d'obtenir une version déterministe en choisissant la direction d égale au sens des aiquilles d'une montre si l'identite est paire, et égale au sens contraire si

l'identite est impaire. Cependant, si les IDs des voisins sont connues, d'autres variantes déterministes sont possibles. Elles different par la phase d'initialisation.

En comparant son ID avec celles de ses voisins, un processus initiateur peut agir selon une ou plusieurs strategies.
La figure suivante énumère quelques unes de ces strategies.



En general, (n>=3) il existe trois cas: le noeud initiateur peut être plus petit que les deux voisins, il peut. être plus grand, et il peut être entre les deux. Pour tous les cas, la stratégie determine le voisin auquel le message sera adressé.
Min* est la version proposée dans [BOD 861, elle suppose que tous les processus sont initiateurs. Un processus n'initialise pas l'algorithme et ne propage pas son message si il sait qu'un de ses voisins a une identité supérieure a la sienne.

Méthode 3: Une autre strategie permettant d'accelerer le processus d'extinction selective de messages(processus) consiste pour un processus donné a adresser directement son IDi au processus dont l'identite IDj est inferieure à IDi. Cette stratégie utilise la notion de sens d'orientation global et nécessite la connexion des processus par d'autres liaisons. La topologie en anneau ne suffit pas a l'application d'une telle strategie. Cette technique est appliquee, par exemple, dans [LOU 861 sur une topologie complete.

Ce qui différencie les algorithmes d'election d'une topologie a une autre c'est l'algorithme de parcours sous-jacent a l'algorithme d'election proprement dit. En faisant abstraction du problème de parcours, les mecanismes utilises pour élire un processus unique sont semblables a ceux appliques a la topologie en anneau. L'utilisation de la technique d'extinction selective a été, en effet, généralisée aux topologies complete et quelconque. Cela s'est fait de deux manieres:

1. La comparaison porte directement sur les identites des processus participant a l'election.

2. La comparaison porte sur des couples (PHi, i) où i designe l'identite du processus Pi, et PHi¿ represente la phase de Pi. Dans ce cas PHi peut avoir plusieurs definitions:
a) PHi represente la taille de l'arborescence en construction, b) elle represente le nombre de racines battues(puisque plusieurs arborescences peuvent être en construction simultanee), c) dans [SKY 871 la phase est definie autrement. Lorsque deux messages(jetons) (PHi, i) et (PHj, j) relatifs à deux processus initiateurs Pi et Pj, respetivement, se rencontrent au niveau d'un processus Pk, tel que PHi=PHj alors, un nouveau jeton (PHk=PHi+1, k) est créé. Les jetons relatifs à Pi et a Pj ne sont pas transmis(ils sont éteints)

2 Election dans les réseaux anonymes

Ce type de reseaux est caracterise par l'absence d'identités(ou d'identites uniques) des processus. Ils sont bases sur une procedure permettant de choisir une identite. Cela constitue la premiere tâche que doit assumer chaque processus dès son activation. 11 s'agit de choisir un entier Si au hasard dans l'intervalle d'entiers [1...d], ou un couple ID=(Si, Ti), tel que Si soit determine comme precedemment. Ti est la valeur d'une certaine variable aleatoire, par exemple le nombre de jetés d'une piece avant l'apparition de face.

Le but principale que 1 'on essaie d'atteidre c'est de réaliser une procedure assurant le choix d'identites uniques avec la plus grande probabilité possible.

A partir de la, les techniques habituelles d'election sont exploitees pour élire un processus. Parfois, au cours de l'algorithme, le choix d'autres identites peut s'avérer necessaire lorsqu' un ou plusieurs processus concurrents ont été dotes de la même identite.

Ce type d'algorithmes réalisant une election avec une probabilité d'erreur même infime ne sont pas interessants dans la pratique. L'election est utilisee dans le but de résoudre un probleme mettant en défaut le fonctionnement d'un systeme, et il ne faudrait pas que la solution apportée conduise elle aussi a un autre problème.

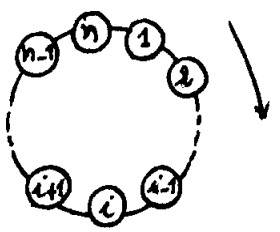
CHAPITRE 4

EVALUATION ET ANALYSE

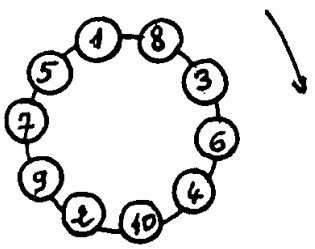
Dans ce chapitre est présentée l'analyse des resultats obtenus par l'evaluation d'un echantillon d'algorithmes. Les performances en nombre de messages et en temps d'élection sont donnees pour chaque algorithmes. Ce chapitre comporte Cgalement une etude comparative des differents algorithmes évalués.

L'evaluation a été réalisée sur les configurations suivantes:

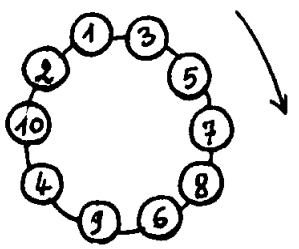
- anneau croissant: les processus sont places de façon ordonnee selon leurs identites croissantes(ordre naturel).



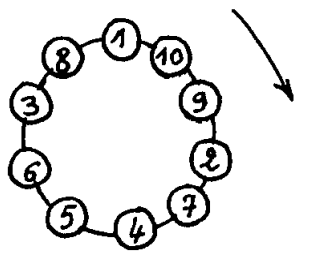
- anneau quelconque: le placement des processus est quelconque.



config. 1

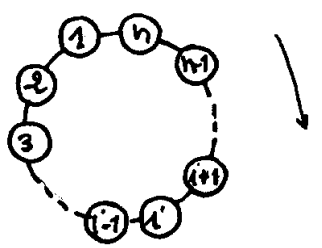


config. 2

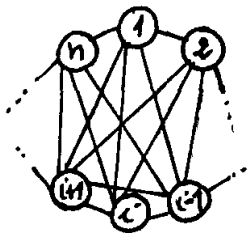


config. 3

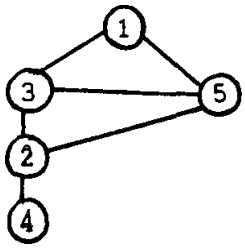
- anneau decroissant: les processus sont places de façon ordonnee selon leurs identites décroissantes.



- reseau complet:



- reseau quelconque:



La representation des resultats a été faite comme suit:

1. L'election collective, en faisant varier la taille n du reseau, permet de representer la variation du nombre de messages ou le temps d'election en fonction de n.

2. L'election collective avec n-10, permet de representer le nombre de messages par site, et/ou le nombre total de messages ainsi que le temps d'election par configuration de reseau.

3. L'election individuelle avec n-10, permet de representer le nombre de messages par site, et/ou le nombre de messages total et le temps d'election en fonction du processus initiateur.

I EVALUATION DES PERFORMANCES

Les resultats relatifs à chaque algorithme sont présentés et analyses dans ce paragraphe.

Les valeurs donnees ici concernent le nombre de messages necessaires à la determination de l'elu, mais chaque processus émet un message de plus (au total n-l messages de plus) pour informer son voisin de l'identite de l'elu. De même, le temps donné est celui mesuré à partir du lancement de l'élection jusqu'à ce que l'identite de l'élu soit connue par un des processus(le premier a conclure).

1 Topologie en anneau

1.1 Algorithmes De Chang Et Roberts [CHA 79]

a) Complexité en messages:

Les bornes **minimale(quand** Pmax lance **l'élection)** et **maximale** prevues par **l'** analyse theorique, respectivement n [figures **IV.6a, IV.6b,** IV. **5a, IV.5b),** et **n(n+1)/2** (figures **IV.4a, IV.4b)** sont les **mêmes** que les **complexités obtenues.**

In	Topologie réseau	mode exéc.	mode élect	Imsges théorique	Imges obtenus	I temps Iobtenu	I Figures
n	Anneau croissant	Asyn. Sync.	col.	O(nLogn)	<=2n-3 2n-1	3n-2 I n	IV.1
10	même	IAsyn Sync.	lindivl P10 I		10 10	16 10	IV.6
10	même	Asyn Sync.	lindivl Pi		11-19 idem	17-39 12-20	IV.5
In	Anneau décrois.	Asyn. Sync.	col.	n(n+1)/2 I	n(0.3n+2) I n(n+1)/2	2n+9 I n	IV.4
1101	même	IAsyn Sync.	lindivl P10	O(nLogn)	10 10	15 10	xv.7 I
10	même	IAsyn Sync.	lindivl Pi I		11-19 idem	I 22-30 I 12-20	IV.7 I
10	Config 1	Asyn. Sync.	col.	O(nLogn)	30 33	21 10	IV.2 I
		IAsyn Sync.	lindivl P10 I		10 10	17 10	IV.6 I
10	Config 2	Asyn. Sync. I	col. I	O(nLogn)	19 22	19 10	IV.2 I
		Asyn Sync.1	lindivl P10		10 10	18 10	IV.6
10	Config 3	Asyn. Sync.I	col.	O(nLogn)	30 33	25 10	IV.2 I
		IAsyn Sync.	lindivl P10 I		10 10	21 10	IV.6

Table IV.1

Pour une election collective sur un anneau croissant, les figures IV.la et IV.lb donnent une complexité de (2n-1) et de (2n-3) respectivement. Pour n égal a 10(figures IV.10a,IV.10b) l'election individuelle génère entre 10 et 19 messages.

Lorsque Pmax(P10) lance l'élection(figures IV.6a, IV.6b), la distribution des messages sur les differents processus est uniforme. Tous les processus emettent le même nombre de messages(1 message de type election: au total 10) pour toutes les configurations. Le message tmis par Pmax est le seul qui circulera sur l'anneau.

Comme on peut le constater(table VI.ll, lorsque l'election est individuelle, des comportements identiques sont obtenus pour les deux modes synchrone et asynchrone, quelque soit la configuration et le processus initiateur.

Dans le cas defavorable oh l'election est collective sur un anneau decroissant, le comportement asynchrone(figure IV.4a) est légèrement meilleur oar rapport au comportement synchrone(figure IV.4b), les complexités obtenues Ctant respectivement n(0.3n+2) et n(n+1)/2.

b) Complexite en temps:

O(n) est la complexité évaluée(figures IV.1c, IV. 1d, IV.4c, IV.4d).

En mode asynchrone le temps enregitre est plus élevé qu'en mode synchrone. Dans ce dernier cas, n unites de temps suffisent pour que le processus Pmax sache qu'il est élu, lorsque l'election est collective(figures IV.ld, IV.4d) ou individuelle initialisée par Pmax(figures IV.6d). Si par contre l'initiateur n'est pas Pmax, ce temps varie entre 12 et 20 unites (figures IV.5d, IV.7d). En mode asynchrone, et toujours pour l'election individuelle, il varie entre 22 et 30 si l'anneau est decroissant (figure IV.7c), et entre 17 et 39 s'il est croissant(figure IV.5c).

c) Conclusion:

L'élection individuelle est plus performante que l'élection collective. Ce résultat est logique car dans le cas d'une election individuelle, seul le processus initiateur Cmet un message et a tout instant, l'extinction selective laissera un seul message en transit sur l'anneau. Dans le cas collectif, initialement tous les processus génèrent un message.

Le comportement asynchrone est meilleur en élection collective, par rapport au comportement synchrone pour ce qui est du nombre de messages générés. Cela peut s'expliquer par le fait qu'en mode asynchrone, les processus evoluent à des vitesses variables, et que les plus lents recevront des messages avant d'en emettre. Les informations transportées par ces messages eviteront aux processus recepteurs d'en emettre

inutilement.

Par contre, le temps d'élection obtenu est meilleur en mode' synchrone, et cela pour les elections collective et individuelle. En effet, en mode synchrone les processus évoluent a la même vitesse ce qui permet de gagner du temps vu que plusieurs messages sont traités en parallele. Ce résultat sera rencontré, par la suite, au niveau de tous les algorithmes.

1.2 Algorithme De Dolev, Klawe Et Rodeh [DOL 82]

In I	Topologie réseau	mode exéc.	lmode élect	lmsges théorique	lmges obtenus	ltemps lobtenu	Figures
In	Anneau croissant	Asyn.	col.	2nLogn+n	3n	>=2n n+l	IV.1 IV.3
		Sync.			3n		
In	Anneau décrois.	Asyn.	col.		3n	<2n	IV.4
		Sync.			3n	n+1	
10	Config 1	Asyn.	col.		50	30	IV.2
I		Sync.			50	16	
10	Config 2	Asyn.	col.			36	IV.2
		Sync.			idem	20	
10	Config 3	Asyn.	col.			33	IV.2
		Sync.			idem	17	

Table IV.2

a) Complexité en messages:

Le nombre de messages obtenu est en 0(n)(figures IV.la, IV.lb, IV.4a, IV.4b). Que l'anneau soit croissant ou decroissant, 3n messages sont générés en mode d'execution synchrone ou asynchrone.

Pour n fixé a 10 processus, 30 messages sont obtenus sur un anneau croissant et sur un anneau decroissant. 50 messages ont été générés sur un anneau où les processus sont places dans un ordre quelconque (figures IV.2a, IV.2b). Ceci montre que le nombre de messages n'est pas sensible au placement des processus sur 1 'anneau, lorsque l'ordre est quelconque.

Le nombre de messages générés par chaque processus est le même(figures IV.3a). Cet algorithme permet de réaliser un équilibre de charge.

b) **Complexité** en temps:

La complexite obtenue est O(n) unites de temps; (n+1) unités en mode **synchrone(figures IV.1d, IV.4d)** et plus de 2n unites en mode **asynchrone(figures IV.1c, IV.1d).** L e temps obtenu pour un anneau croissant est exactement le même que pour un anneau **décroissant** en mode synchrone.
Pour les autres configurations, sont enregistres 16, 20 et 17 unites de temps en mode synchrone, et 30, 36 et 33 unites de temps en mode asynchroneffigure **IV.3b).**

cl Conclusion:

Le **comportement** de l'algorithme est le **même** pour les deux modes d'execution synchrone et asynchrone du point de vue nombre de messages. C'est-a-dire que le nombre de messages ne depend pas de la vitesse **d'exécution** des differents processus. Par **contre**, du point de vue temps, le mode synchrone est plus performant, comme **c'est** le cas pour l'algorithme de Chang [CHA 791 ci-dessus.

1.3 Algorithme De Hirshbeg Et Sinclair [HIR **80]**

a] **Complexité** en messages:

Dans les figures IV.la et **IV.1b**, la variation du nombre de messages en fonction du nombre de processus composant le reseau, est une fonction **croissante.** Elle est caracterisee **par** des augmentations brusques qui se traduisent au niveau des figures **IV.1a**, IV.lb et IV.le par une augmentation nette du nombre de messages. Ce **phénomène** peut **être** explique par le fonctionnement de l'algorithme **lui-même. .En** effet, l'algorithme **réitère** sur des chemins de longeur 2^i, i=0,1,2,...k, jusqu'à atteindre une longeur 2^k **pouvant** contenir **tout** l'anneau.

Il est facile de verifier que pour des reseaux dont la taille n, est comprise entre 2^{k-1} et 2^k, c'est-a-dire $n \in \{2^{k-1}+1, 2^{k-1}+2, \ldots 2^k\}$ les iterations porteront sur des chemins de longeur $1,2,4,\ldots 2^k$. Lorsque la taille du réseau prend la valeur 2^k+j, $j \in \{1,2,3,..\}$, tel que $2^k+j<2^{k+1}$, les iterations iront jusqu'à 2^{k+1}. Autrement dit, pour $2^{k-1}<n<2^k$ la longeur maximale atteinte sera toujours 2^k, et **cela** se traduit par une demi-droite au niveau des figures **IV.1a, IV.1b** et IV.le. Des que n prend une valeur superieure a 2^k, l'algorithme itere sur un chemin de longeur **maximale égale à** 2^{k+1}, et **cela** se traduit par le passage a la demi-droite superieure. Les demi-droites sont paralleles; elles ont **donc** le **même** coefficient directeur. Les equations qui les caracterisent sont de la **forme** an+bi, où bi varie d'une demi-droite **à l'autre(voir** la table **IV.3).**

Rappelons **que** les processus sont places dans l'ordre nature1 de leurs **identités,** ce qui pourrait expliquer que la complexite trouvee **soit** differente de **celle** qui **était** estimee **par** l'analyse **théorique(où** la configuration n'a **pas été précisée).** D'ailleurs, le nombre de messages est plus **élevé**

quand les procesus sont places dans un ordre quelconque(table IV.3, et les figures A.1a, A.1b).

Lorsque l'élection individuelle est lancee par Pmax, le nombre de messages engendres est le même pour toutes les configurations de l'anneau(figures IV.6a, IV.6b). De plus, le mode d'execution n'a pas influence cette complexite.
L'élection individuelle lancee par un processus different de Pmax est représentée sur les f iqures IV.lOa et IV.10b. La complexite dans ce cas, varie entre 84 et 111 messages, et est pratiquement la même en mode synchrone ou asynchrone.

b) **Complexité** en temps:

L'evolution du temps d'election en fonction de la taille du réseau suit le même processus que le nombre de messages. Elle est lie directement au fonctionnement de l'algorithme.

In I	Topologie réseau	mode exéc.	mode élect	msges théorique	mges obtenus	temps obtenu	Figures
In	Anneau lcroissantl	Asyn. Sync.	col.	8n+8nLogn ~O(nLogn)	$4n+4\sum_{i=1}^{\lfloor \log n \rfloor} 2^i$ $6n+4\sum_{i=1}^{\lfloor \log n \rfloor} 2^i$	même allure	IV.1
10	même	Asyn Sync.	lindivl P10		79 80	81 40	IV.6
10	même	Asyn Sync.	lindivl Pi		84-111 idem	81-98 48-57	IV.5
10	Config 1	Asyn. Sync.	col.	O(nLogn)	143 116	84 50	IV.2
		Asyn Sync.	lindivl P10		80 80	84 48	IV.6
10	Config 2	Asyn. Sync.	col.	O(nLogn)	127 132	84 51	IV.2
		Asyn Sync.	lindivl P10		79 80	89 48	IV.6
10	Config 3	Asyn. Sync.	col.	O(nLogn)	128 140	76 49	IV.2
		Asyn Sync.	lindivl P10		79 80	76 48	IV.6

Table IV.3

Contrairement à ce qui a été observe pour le trafic, le comportement temporel en mode synchrone est meilleur par rapport au comportement en mode **asynchrone**(figures IV.1c et IV.1d). Pour un anneau de 10 processus de configuration quelconque (figure IV.2d) le temps d'election obtenu varie entre 48 et 51 unites en mode synchrone, et entre 76 et 84 unites en mode asynchrone (figures IV.2c).

Le temps enregistre pour une election individuelle est donne dans les figures IV.5c, IV.5d, IV.6c, IV.6d ainsi que la table IV.3.

Les figures IV.5c, IV.5d montrent que le temps d'election ne varie pas **beaucoup** lorsque l'initiateur change; car les messages sont émis dans les deux directions, et le comportement est symetrique de part et d'autre de cet initiateur quelqu'il soit.

cl Conclusion:

L'election individuelle engendre moins de messages que l'election collective. Le nombre de messages est plus **élevé** en mode synchrone et **cela** pour l'election collective. Le temps est. plus élevé en mode asynchrone quelque soit le type d'election. L'explication de ce résultat rejoint celle donnee pour l'algorithme [CHA 791.

1.4 Algorithme-P De Korach, **Rotem** Et Santoro **[KOR 81]**

L'algorithme décrit dans [KOR 81] suppose que tous les processus initialisent simultanement l'algorithme. Sinon, a la reception du premier message, un processus Cmet un message, contenant son identité, avant d'analyser le message reçu.

Cet algorithme a été modifié de façon a permettre à un processus d'analyser le contenu du message reçu avant de transmettre *son* propre message(la version modifiée de l'algorithme a été designee par "Korach-2"). Cette modification a été introduite dans le but d'éviter l'emission inutile d'un message dans le cas où l'identite transportee par le message reçu est superieure à celle du processus recepteur. Autrement dit, un processus génère son propre message seulement si son identité est superieure à celle contenue dans le message reçu.

a) Complexité en messages:

En ce qui concerne l'algorithme de Korach [KOR 81], la complexité obtenue dans le cas d'une election collective, est O(nLogn+n). Pour n égal à 10, le nombre de messages obtenu est au plus égal à 51 pour tous les types d'election et de configurations confondus(table IV.41. En mode synchrone, le nombre maximum de messages est égal à 40.

La comparaison des deux versions de l'algorithme, la version d'origine [KOR 811 et la version modifiée Korach-2 permet de conclure que: pour l'election collective, les deux algorithmes ont presque les **mêmes complexités** en mode **asynchrone(figure IV.1a),** et exactement la **même** en execution synchrone (figure **IV.1b).** Ces observations se generalisent au cas d'un anneau quelconque(figures **IV.2a, IV.2b,** table IV.4).

En elect ion individuelle, les figures **IV.5a** e t IV. 5b montrent que la version Korach-2 offre la meilleure **complexité(cela** se verifie egalement **sur les** f iqures **IV.6a, IV.6b).** Dans le cas des figures **IV.5a** et **IV.5b,** le **nombre** de **messages varie entre 23 et** 51 **en asynchrone, et entre** 22 **et** 40 en synchrone pour l'algorithme **[KOR 81].** 11 varie entre 11 et 20 et entre 11 et 22 pour Korach-2.

1. Algorithme de [KOR 81]

In	Topologie reseau	mode exéc.	lmode élect	lmsges théorique	mges obtenus	temps obtenu	Figures
In	Anneau croissant	Asyn. Sync.	col.	$n^2/2$	O(nLogn+n)	2n+2 n	IV.1
10	même	Asyn Sync.	indiv P10	(3n Hn)/4	38 34	31 13	IV.6
10	même	Asyn Sync.	lindivl Pi		23-51 22-40	23-36 13-19	IV.5
10	Config 1	Asyn. Sync.	col.	O(nLogn)	25 26	19 11	IV.2
		Asyn Sync.	lindivl P10		20 20	27 12	IV.6
10	Config 2	Asyn. Sync.	col.	O(nLogn)	31 22	16 11	IV.2
		Asyn Sync.	lindivl P10		34 26	31 12	IV.6
10	Config 3	Asyn. Sync.	col.	O(nLogn)	27 27	20 11	IV.2
		Asyn Sync.	lindivl P10		19 25	21 11	IV.5

Table IV.4

b) Complexité en temps:

L'agorithme de Korach [KOR 811 a une complexite de 2n unites en mode asynchrone et environ n en mode synchrone, pour l'election collective sur un anneau croissant(figures IV.1c, IV.1d). La version Korach-2 presente la même complexite que la version d'origine [KOR 811 en mode synchrone, et une légère difference en mode asynchrone.

Lorsque n est fixé à 10 processus, et en faisant varier le placement des processus sur l'anneau, les résultats obtenus sont:

- Dans l'election collective en mode asynchrone(figures IV.2c), sont utilisées pour [KOR 811 entre 16 et 20 unites de temps, et pour Korach-2 entre 19 et 23 unites . En mode synchrone 11 unites de temps ont été enregistrees pour les deux algorithmes(figures IV.2d).

2. Algorithme de Korach **modifié:** Korach-2

In	Topologie réseau	mode exéc.	mode élect	mges obtenu	temps	Figures
In	Anneau croissant	Asyn. Sync.	col.	O(nLogn+n)	2n n	IV.1
10	même	Asyn Sync.1	lindivl P10	10 10	14 10	IV.6
1101	même	Asyn Sync.	lindivl Pi	11-20 11-22	17-32 11-19	IV.5
10	Config 1	Asyn. Sync.	col.	25 2	19 6	IV.2 11
		Asyn Sync.	lindivl P10	10 10	18 10	IV.6
1101	Config 2	Asyn. Sync.	col.	34 22	16 11	IV.2
		Asyn Sync.	lindivl P10	10 10	21 10	IV.6
10	Config 3	Asyn. Sync.	col.	27 27	20 11	IV.2
		Asyn Sync.	lindivl P10	10 10	21 10	IV.6

Table **IV.4(suite)**

- L' election individuelle initialisee par Pmax, donne pour [KOR 81] entre 21 et 31 unites de temps en mode asynchrone, et entre 18 et 21 unites pour Korach-2(figures IV.5c, IV.6c). En mode synchrone(figures IV.5d, IV.6d) entre 11 et 12 unites sont obtenues pour [KOR 81], et 10 unites pour Korach-2 quelque soit la configuration.

L'election individuelle initialisee par un processus Pi autre que Pmax donne pour Korach-2 de meilleurs résultats que pour [KOR 81] en mode d'execution synchrone et asynchrone(figures IV.5c, IV.5d).

c) Conclusion:

Les deux versions CKOR 811 et Korach-2 ont les mêmes comportements temporels et en nombre de messages quand l'élection est collective. L'alqorithme Korach-2 est plus performant en election individuelle. Comme pour les algorithmes precedents, le mode synchrone offre le meilleur temps d'élection.

2 Topoloqie complete

2.1 Algorithmes De Afek Et Gafni [AFE 851

Les trois algorithmes proposes A, B et C tentent chacun d'améliorer les performances de l' autre. L'algorithme B est conçu pour ameliorer la complexite en messages de A, et l'algorithme C est conçu de façon a preserver la complexite en messages de B et a en ameliorer la complexite temporelle.

2.1.1 L'algorithme A

In	Topologie réseau	mode exéc.	mode elect	msges theorique	lmges lobtenus	temps lobtenu	Figures
In	Reseau complet	Asyn. Sync.	col.	2.77nLogn	3n+1 3n	3n+6 2n+1	IV.8
10		Asyn. Sync.	col.	92	32 31	33 21	IV.8 IV.9
10		Asyn. Sync.1	indiv P10		20 20	36 20	IV.10 IV.11
10		Asyn. Sync.	indiv Pi		20 20	35-43 20-20	IV.10

Table IV.5

a) Complexité en messages:

L'election collective a une complexité(figures IV.8a, IV.8b) de 3n messages en mode synchrne et de 3n+1 en mode asynchrone. Avec n-10, 32 messages sont enregistres pour le cas synchrone et 31 pour le cas asynchrone.

En faisant varier le processus initiateur, pour une election individuelle, les figures IV.10a et IV.10b donnent le même nombre de messages(20) émis au total dans chaque cas et pour les deux modes d'execution.

b) Complexité en temps:

Elle est situee autour de la valeur de 2n pour l'election collective synchrone(Figure IV.8d) et de 3n pour l'election collective asynchrone(figure IV.8c).

En execution asynchrone 49 unites suffisent pour la terminaison de l'algorithme dans une election collective(figure IV.9c). Le temps mesuré varie entre 35 et 43 unites pour l'election individuelle(figure IV.10c). En execution synchrone 31 unites de temps sont necessaires pour 1 'election. collective(figure IV.2c), et, 20 uni tds pour l'election individuelle quelque soit le processus initiateur(IV.10d).

2.1.2 L'algorithme B

In I	Topologie réseau	mode exéc.	lmode élect	lmsges théorique	mges obtenus	temps lobtenu	Figures
In I	Réseau complet	Asyn. Sync.	col.	2nLogn+2n	3n+5 3n+4	2.8n+25 2n+8	IV.8
1101 I		Asyn. Sync.	col.	86	35 34	49 31	IV.8 IV.9
10 I		Asyn. Sync.	indivl P10		20 20	36 20	IV.10 IV.11
10 I		Asyn. Sync.1	indiv Pi		20 20	35-43 20-20	IV.10

Table IV.6

a) Complexité en messages:

La complexite évaluée est O(n) (figures IV.8a et IV.8b), elle est donc inferieure à la complexite theorique de 2nLogn+2n. Avec n-10, le nombre de messages obtenu est égal à 35 en mode asynchrone, et a 34 en mode synchrone(figure IV.91.

Comme l'algorithme A, l'election individuelle(figures IV.lOa, IV.10b) nécessite 20 messages quelque soit l'initiateur et le mode d'execution.

b) Complexité en temps:

Le temps est mesuré par O(n) unites. Environ 3n unites pour le cas asynchrone et 2n unites pour le cas synchrone ont été enregistrées(figures IV.8c et IV.8d respectivementl. La figure IV.8c en mode asynchrone montre que l'algorithme B est plus lent que **l'algorithme A pour une élection** collective. L'election individuelle, par contre est la même pour les deux algorithmes A et B(figures IV.10c, IV.10d, IV.11c et IV.11d).

2.1.3 L'algorithme C

a) Complexité en messages:

L'évaluation donne une complexite d'environ 2.4n pour l'election collective (figures IV.8a et IV.8b). Mais, les trois algorithmes A, B et C ont pratiquement les mêmes complexités.

In I I	Topologie reseau	mode exéc.	lmode élect	lmsges théorique	lmges obtenus	ltemps lobtenu	Figures
In	Réseau complet	Asyn. Sync.	col.	~O(nLogn)	3n 3n+4	2.4n+32 2n+8	IV.8
110		Asyn. Sync.	col.	141	30 34	28 31	IV.8 IV.9
10		Asyn. Sync.	indiv P10		20 20	36 20	IV.10 IV.11
110 I I		Asyn. Sync.	indiv Pi		20 20	35-43 20-20	IV.10

Table IV.7

Le nombre de messages total générés par l'algorithme dans une election individuelle initialisee par l'un quelconque des processus du reseau est Cgal a 20(même que A et B), quelque soit le processus initiateur(figures IV.lOa et IV.10b).

b) Complexité en temps:

Les figures IV.8c et IV.8d donnent la variation du temps d'election en fonction du nombre de processus du reseau. 11 en ressort que la complexite de l'algorithme C est la même que celle de B, en mode synchrone. En mode asynchrone, la valeur moyenne du temps d'election de l'algorithme C est legerement moins &levee que celle de B.

Les figures IV.10c et IV.10d représentent le temps d'election enregistre pour des elections individuelles initialisees par les processus du reseau. Elles montrent que dans le cas synchrone 20 unites sont necessaires quelque soit le processus initiateur. Pour le mode asynchrone le temps varie entre 35 et 43 unites.

c) Conclusion:

Pour les trois algorithmes, la conclusion qui s'impose est que le mode d'execution n'a pas d'effet sur les complexités en trafic engendre. 11 faut noter que pour une election collective en mode asynchrone, ou synchrone les trois algorithmes A, B et C génèrent le même nombre de messages(figures IV.8a, IV.8b). L'election individuelle(figures IV.l0a et IV.10b) est la même pour les trois algorithmes(20 messages obtenus partoutl.

D'après ces résultats, le but fixe dans [AFE 851 et qui consite a ameliorer la complexite en messages de B par rapport à celle de A n'est pas atteint.

Comme cela a été prévu, du point de vue temps d'election, l'algorithme A est le plus performant. Cela est visible dans le cas de l'election collective synchrone(figures IV.8c et IV.8d). Les algorithmes B et C ont la même complexite. L'amelioration de C par rapport a B, n'est réalisée (figure IV.8c),mais tres faiblement, que dans l'élection collective asynchrone.

2.2 Algorithme De Loui, Hatsushita Et West [LOU 861

a) Complexite en messages:

Le nombre de messages est égal a environ 3n en mode d'execution synchrone ou asynchrone pour l'élection collective(figures IV.8a et IV.8b).

Quand Pmax lance l'élection(figure IV.lll, le nombre de messages émis par chaque processus, sauf Pmax, est le même. En faisant varier le processus initiateur(figure IV.10a, IV.10b) le nombre total de messages générés par l'algorithme semble en être très peu affecté; la valeur moyenne est Cgale a 29. Cela peut s'expliquer par l'utilisation du sens d'orientation global qui permet un adressage direct des messages.

b) Complexité en temps:

En mode synchrone, le temps d'election en fonction de la taille du réseau est égal à n+1(figure IV.8d). E n mode asynchrone ce temps est presque constant, c'est-a-dire qu'il varie autour d'une moyenne de 22 unités(figure IV.8c).

L'election individuelle initialisee par les differents processus, aboutit a un temps variant entre 9 et 12 unites pour le mode synchrone (figure IV.10d). Par contre en mode asynchrone ces temps varient en fonction de l'initiateur entre

In	Topologie reseau	mode exéc.	mode élect	msges théorique (total)	mges obtenus	temps obtenu	Figures
n	Réseau complet	Asyn. Sync.	col.	3.62n	3n 3n	<=25 n+1	IV.8
10		Asyn. Sync.	col.	36	28 29	19 11	IV.8 IV.9
10		Asyn. Sync.	indiv P10		32 29	25 12	IV.10 IV.11
110		Asyn. Sync.	indiv Pi		27-32 28-29	15-30 9-12	IV.10

Table IV.8

15 et 30 unités(figure IV.10c).

c) Conclusion:

Cet l'algorithme est plus performant, du point de vue temps, que les algorithmes A, B et C précédents(figures IV.8c et IV.8d).

3 Topologie Quelconque

3.1 Algorithme De Helary, Naddi Et Raynal [HEL 86bl
(exploration en profondeurl

A/ Topologie en anneau

a) Complexité en messages:

Le nombre de messages obtenu pour une election collective synchrone ou asynchrone, sur un anneau croissant ou decroissant, est de l'ordre de O(n)(figures IV.12a, IV.12b). Par exemple, pour 10 processus 18 messages sont obtenus.

Pour une election individuelle, lorsque le processus initiateur varie, le nombre de messages se situe entre 10 et 17(figures IV.13a et IV.13b).

Le placement des processus de façon quelconque sur le réseau(figures IV.2, IV.2) fait varier le nombre de messages entre 24 et 33 messages pour l'élection collective.

Dans les deux modes d'élection collective et individuelle, le comportement observe est le même pour une exécution synchrone et asynchrone.

en mode asynchrone et entre 10 et 18 unités en mode synchrone.

En election individuelle(figures IV.5c et IV.5d) le temps varie entre 15 et 29 unites selon l'initiateur en mode asynchrone et, entre 11 et 18 unites en mode synchrone.

Comme pour les autres algorithmes, le comportement synchrone est plus performant que le comportement asynchrone.

B/ Topologie complete

a) Complexite en messages:

L'évaluation a aboutit a une complexite de 2n-5 messages(figures IV.12a, IV.12b). C'est-a-dire 3(n-1) messages son obtenus en comptant les messages de conclusion, contre (n-1)(n-3) prévus par l'analyse theorique dans le cas d'une election collective(n=10, 27 messages, contre 631.

Dans le cas d'une election individuelle où Pmax est seul a lancer l'election, on prévoi 2(n-1) messages au total, et c'est exactement la valeur qui a été obtenue(figures IV.11a et IV.11b). Quand P10 initialise l'election, 9 messages sont générés(sans compter les messages de conclusion). Si un processus autre que P10 lance l'election, 10 messages sont générés.

Le mode d'execution n'a pas influence6 l'election: les resultats recueillis dans les deux cas sont semblables, cela est vrai pour les elections collective et individuelle.

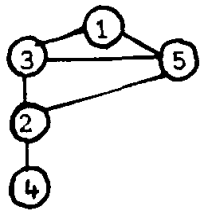
b) Complexite en temps:

Lorsque l'election est collective la complexite obtenue pour le mode synchrone et le mode asynchrone est O(n)(figures IV.12c et IV.12d). Pour n=10, 22 unites sont obtenues en mode asynchrone et 10 en mode synchrone (figure IV.91.

Si l'election est individuelle, le temps mesuré vaut 15 unités(pour la plupart des cas) ou 21 unites selon l'initiateur(figures IV.14c, VI.14d) en mode asynchrone. Il vaut 18 unites en mode synchrone avec Pmax comme seul initiateur (figure IV.11), et 19 unites avec les autres processus initiateurs.
Le mode synchrone donne le mellleur temps.

C/ Topologie quelconque

L'étude a porté sur le reseau suivant:



a) Complexité en messages:

Pour l'election collective (figures IV.15a, IV.15b), le nombre de messages est egal a 11 en execution synchrone, à 9 messages en execution asynchrone. Un trafic plus important est observe au niveau des processus P2 et, P3 a cause de leur position relais (ils font suivre les messages en provenance des processus P4 et P5.

Si l'election est individuelle, le nombre de messages varie entre 8 et 9 selon que c'est Pmax ou un autre processus qui lance l'élection (figures IV.16a, IV.16b). On prevoit 12 messages au maximum quand Pmax initialise l'election, dans [HEL 86b].

b) Complexité en temps:

Avec cette configuration de cinq pr. sus, les résultats sont:

- pour l'election collective asynchrone 11 unites et pour le cas synchrone 7 unités (figur IV.15c) sont nécessaires.

- pour l'election individuelle asynchrone 10 unites sont enregistrées (figures IV.16c). En mode synchrone le temps obtenu varie entre 8 et 9 unites, et est égale à 7 unites lorsque l'initiateur est Pmax (figures IV.16d).

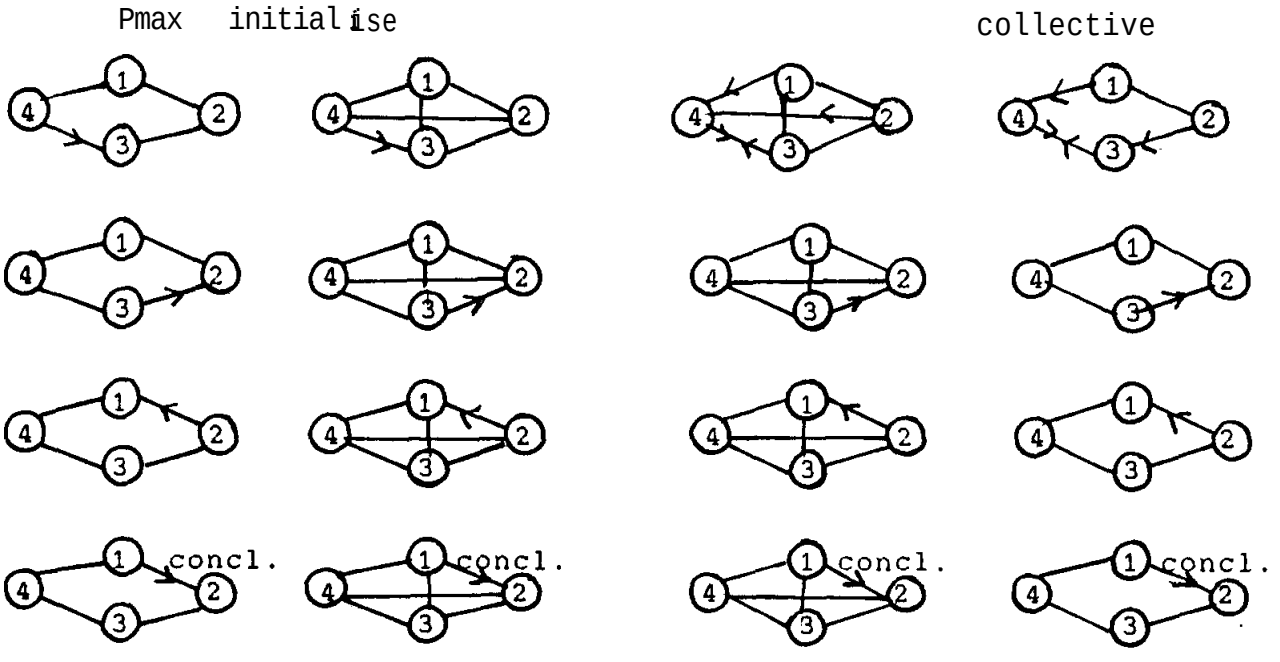
c) Conclusion

En resume, pour les performances en trafic, le mode asynchrone et le mode synchrone donnent les mêmes complexites. Mais, en temps, le mode synchrone est mieux que le mode asynchrone (l'explication a été donnee precedemment pour l'algorithme [CHA 791).

Pour l'exploration en profondeur, que le réseau soit en anneau ou à maillage complet, les complexités obtenues en messages (figures IV.1a, IV.1b) et en temps (figures IV.1c, IV.1d) sont les mêmes pour les deux modes d'exécution, l'blection étant collective. Cette constatation est également vraie pour une election individuelle initialisée par Pmax (figures IV.13a, IV.13b, IV.14a, IV.14b)

Ce phénomène peut être explique en regardant de plus près le fonctionnement de l'algorithme. En effet, comme le message est toujours envoyé vers le voisin dont l'identite est maximale, il sera toujours amené à circuler sur un anneau.

Soit l'exemple suivant:



Ces deux exemples montrent bien qu'il y a égalité de performances pour les deux topologies en anneau et a maillage complet.

3.2 Algorithme De Helary, Maddi Et Raynal [HEL 86b]

(exploration en largeurl

A/ Topologie en anneau

al Complexité en messages:

La courbe representant l'evolution du nombre, de messages en fonction du nombre de processus(figures IV.12a, IV.12b) est de la forme O(nLogn), pour les executions synchrone et asynchrone. Pour n=10, le... nombre de messages est egal a 49 en execution asynchrone et a 56 en execution synchrone(figures IV.2a, IV.2b).

Des configurations oh le placement des processus est quelconque(figures IV.2a, IV.2b) ont été étudiées pour une election collective, et les complexiés qui en ont , été deduites varient entre 45 et 47 messages en mode asynchrone et, entre 47 et 53 en mode synchrone.

En faisant varier l'initiateur, le nombre de messages varie entre 22 et 53 messages et entre 22 et 55 messages, pour respectivement un mode asynchrone ou synchrone. Ces résultats correspondent au cas où les processus sont placés dans l'ordre

croissant ou decroissant de leurs identités(figures IV. 13a, IV.13b).

In	Topologie réseau	mode exec.	lmode élect	lmsges théorique	Imges obtenus	Itemps obtenu	Figures
i n	Anneau croissant Idecrois.	Asyn. Sync.	col.		O(nLogn+n) O(nLogn+n)	1.7n+7 1.2n+5	IV.12
10		Asyn Sync.	lindivl P10 I		22 22	23 13	IV.13
10		Asyn Sync.	lindivl Pi I		22-53 22-55	29-37 14-18	IV.13
IlOlConfig 1	anneau	Asyn. Sync.	col.		45 47	29 14	IV.2
IlOlConfig 2	anneau	Asyn. Sync.	col.		47 53	22 14	IV.2
IlOlConfig 3	anneau	Asyn. Sync.	col.		46 51	32 15	IV.2
n	réseau complet	Asyn. Sync.	col.	$O(n^2)$	$O(n^2)$ $O(n^2)$	2n+9 2n-1	IV.8
10		Asyn Sync.	lindivl P10 I		18 18	16 10	IV.14
10		Asyn Sync.	lindivl Pi		18-80 18-108	15-28 11-18	IV.14
5	réseau quelconq.	Asyn. Sync.	col.		11 17	7 7	IV.15
5		Asyn Sync.	lindivl P5		8 8	6 5	IV.16
15		Asyn Sync.	lindivl Pi		8-14 8-15	9-11 6-7	IV. 16

Table IV.10

b)Complexité en temps:

L'evaluation donne une variation lineaire du temps d'election en fonction du nombre de processus(figures IV.12c, IV.12d). Pour 10 processus, il faut 24 unites de temps pour que l'identite de l'élu soit connue,en execution asynchrone(figure IV.11c), tandis que 14 unites sont necessaires en mode

synchrone(figure IV.11d).

Pour l'election individuelle, en faisant varier l'initiateur, le temps d'election varie entre 29 et 37 unites, et entre 14 et 18 unites, le mode d'execution étant respectivement, asynchrone(figure IV.13c) puis synchrone(IV.13d). A l'inverse de la complexité en messages qui est un peu plus élevée dans le mode synchrone, le temps d'election est bien plus petit dans le mode synchrone par rapport au mode asynchrone.

B/ Topologie complete

a) Complexité en messages:

La complexite obtenue(figures IV.12a, IV.12b) est de l'ordre de n^2, pour l'election collective.

L'election individuelle a donné 18 messages quand Pmax initialise l'élection(figure IV.lll. Le nombre de messages varie entre 18 et 80 quand un processus Pi, autre que P10, initialise l'election pour le mode asynchrone (figure IV.14a), et entre 18 et 108 en mode synchrone(IV.14b).

b) Complexité en temps:

Pour une election collective(figures IV.12c, IV.12d), le temps mesuré est O(n): 26 unites en mode asynchrone et 19 unites pour le cas synchrone dans un réseau de taille n=10(figure IV.9).

Pour une election individuelle, le temps varie en fonction du processus lancant l'election. Quand ce dernier est Pmax, 16 et 10 unites sont obtenues pour les modes asynchrone(figure IV.14c) et synchroneffigure IV.14d). Dans les autres cas le temps varie entre 15 et 28 unites, et entre 11 et 18 unites pour respectivement des executions asynchrone(figure IV.14c) et synchroneffigure IV.14d).

C/ Topologie quelconque

a) Complexité en messages:

Les figures IV.15a et IV.15b montrent que le nombre de messages est égal à 11 en mode asynchrone, et a 17 en mode synchrone pour une election collective.

Un trafic plus important est observe au niveau des processus 2, 3 et 5. Ces derniers ont un plus grand nombre de voisins par rapport aux processus 1 et 4. Pour une election individuelle initialisée par Pmax ou par les autres processus a tour de rôle(figures IV.16a, IV.16b), on obtient entre 8 et 14 messages en mode asynchrone, et entre 8 et 15 messages en mode synchrone.

b) Complexite en temps:

Pour une élection collective 7 unites de temps sont obtenues en mode asynchrone(figure IV.15c), et en mode synchrone(figure IV.15d).

Pour l'election individuelle, entre 9 et 11 unites ont été enreqistrees en mode asynchrone(figure IV.16c), 6 unites quand Pmax est l'initiateur. Dans le cas synchrone, le temps varie entre 6 et 7 unités(figure IV.16d), 5 unites sont enregistrées quand Pmax est l'initiateur.

3.3 Comparaison des explorations en profondeur et en largeur

3.3.1 Election collective

al Comparaison des trafics

Reseau en anneau: L'exploration en profondeur donne de meilleurs résultats (figures IV.12a, IV.12b) en modes synchrone et asynchrone, par rapport à l'exploration en largeur. Ceci est previsible puisque dans le premier algorithme, un processus donne emet un message d'exploration vers un seul de ses voisins(celui dont l'identite est maximale), tandis que dans le second, il diffuse le message d'exploration a l'ensemble de ses voisins. Pour n=10, 27 messages sont émis par une exploration en profondeur(synchrone et asynchrone), 49 messages(asynchrone) et 56 messages(synchrone) par une exploration en largeur. Ce résultat est vérifié quelque soit la configuration de l'anneau(figures IV.2a, IV.2b).

Reseau complet: La même conclusion que le cas d'un reseau en anneau(figures IV.12a, IV.12b) peut être répétée ici.

Réseau quelconque: C'est aussi l'exploration en profondeur qui est la plus performante (figures IV.15a et IV.15b).

b) Comparaison des temps d'election

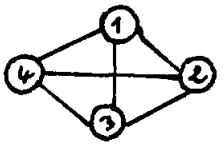
Dans le cas de l'election collective, le temps d'election obtenu dans une exploration en profondeur est plus petit que celui de l'exploration en largeur, et cela pour les topologies en anneau et à maillage complet, en execution synchrone ou asynchrone. 11 faut, cependant, noter que la difference entre les deux explorations est plus importante dans le cas du maillage complet (figures IV.12C et IV.12d). La figure IV.2 confirme ce résultat sur les differentes configuration de l'anneau.

Pourtant, le lancement simultane de l'exploration sur plusieurs chemins, en parallele, pourrait laisser s'attendre à l'acceleration du calcul dans l'exploration en largeur par rapport à l'exploration en profondeur.

Ce résultat peut être explique par le fait que la multiplication excessive des messages se traduit par le temps pris dans le traitement de certains d'entre eux véhiculant des informations sans intérêt et ne permettant pas la conclusion.

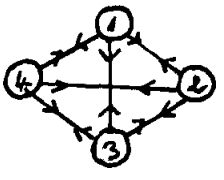
L'exemple de l'élection collective ci-dessous illustre cette situation:

Soit le graphe complet:

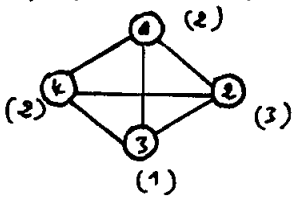


L'exploration en largeur donne les étapes de traitement suivantes:

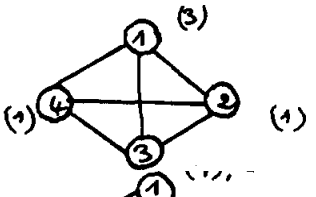
Etape 1: lancement de l'exploration: diffusion a l'ensemble des voisins.



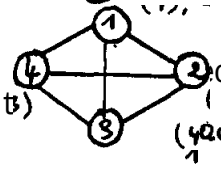
Etape 2: le processus 4 reçoit les messages (1), (2) ou (3). Les processus 1, 2 et 3 recoivent des messages, autres que le message (4), qu'ils acquittent ou eteignent.



Etape 3: comme l'étape 2.



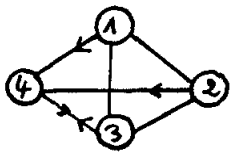
Etape 4: les processus 1, 2 et 3 recoivent chacun le message(4) qu'ils acquittent.



Etape 5: le processus 4 reçoit les acquittements et conclut.

L'exploration en profoncleur donne les etapes suivantes:

Etape 1: chaque processus lance une exploration vers son plus grand voisin.



Etape 2: chaque processus recoit le message qui lui est adresse: le processus 3 recoit (4) qu'il transmet vers 2, le processus 4 recoit (1), (2) ou (3) qu'il eteint.

Etape 3: le processus 2 recoit (4) qu'il transmet vers 1.

Etape 4: le processus 1 recoit le messages (4) et conclut.

Alors que dans l'exploration en profondeur le message émis par le processus P4 est traité par le processus P3 dès l'étape 2, dans l'exploration en largeur il n'est reçu par les destinataires qu'à l'étape 4.

Contrairement au topologies complete et a maillage complet, le cas quelconque étudié(figure IV. 15c) montre que les deux explorations donnent le même temps d'election pour le cas synchrone, cependant, c'est l'exploration en largeur qui offre le meileur temps en mode asynchrone.

3.3.2 Election individuelle

a) Comparaison des trafics

Réseau en anneau: Les figures IV.13a et IV. 13b indiquent que l'exploration en profondeur nécessite moins de messages par rapport à l'exploration en largeur.

Réseau complet: Même remarques que le cas de l'anneau(figures IV.14c et IV.14d).

Réseau quelconque: Les figures IV.16a, IV.16b, montrent également que l'exploration en profondeur est plus performante que l'exploration en largeur.

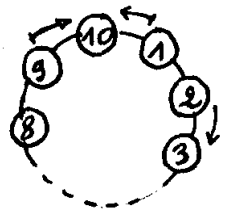
b) Comparaison des temps d'élection

Dans le cas de l'élection individuelle sur une topologie complète(figures IV.14c et IV.14d), l'écart(en temps) entre les deux explorations est d'autant plus reduit que le processus initiateur a uneidentité qui s'approche de la valeur Max(10). Les cas extremes sont:

a) P1 lance l'exploration en diffusant son message (1). A la reception du message (1), chacun des processus diffuse son propre message(P1 ayant l'identite minimale).

b) P10 lance l'exploration, en diffusant le messages (10). A la reception de ce message, tous les processus l'acquittent immediatement sans generer d'autre messages d'exploration.

Sur une topologie en anneau(figure IV.13c et IV.13d) c'est aussi l'exploration en profondeur qui est plus performante et plus particulierement en mode asynchrone. Dans ce cas, il faut remarquer que(surtout en mode synchrone) la difference est moins élevée lorsque l'initiateur est plus près de P10. Cela est dû au fait que l'exploration en profondeur est d'autant plus lente que le processus initiateur s'eloignent de P10. Le meilleur temps est réalisé quand P10, P9 ou P1 sont initiateurs.



La topologie quelconque étudiée(figure IV.16c et IV.16d) met en evidence la rapidite de l'exploration en profondeur en mode asynchrone, et la rapidite de l'exploration en larqeur en mode synchrone.

4.3 Conclusion

La **multiplication excessive des messages dans** le **but d'accélerer la conclusion, peut avoir un effet inverse à cause du temps pris dans le traitement de messages inutiles.** Cela est confirmé par le fait que c'est en election collective, et plus particulierement dans la topoloqie à maillaqe complet, que l'exploration en profondeur est nettement plus rapide que l'exploration en larqeur. Dans cette topoloqie la multiplication des messages est maximale du fait que chaque processus est relié a tous les autres processus du reseau. Par contre, la configuration quelconque étudiée, a montre que l'exploration en larqeur a une meilleure performance temporelle que l'exploration en profondeur(figures IV.15C, IV.16d).

II ETUDE COMPARATIVE GENERALE

Une etude comparative des alqorithes évalués est présentée dans le present paragraphe. Une classification des alqorithmes selon leurs performances est proposée.

1 Topologie En Anneau

Les algorithmes conçus pour la topologie en anneau [CHA 79, DOL 82, HIR 80, KOR 81, Korach-2] sont compares avec les algorithmes d'exploration en largeur et profondeur [HEL 86bl executes sur la même topologie.

1.1 Election collective

a) Comparaison des trafics

La comparaison des algorithmes pour une election collective en mode asynchrone est representee dans la figure IV.1a, et en mode synchrone dans la figure IV.1b. L'alqorithme base sur l'exploration en profondeur a le même comportement que l'algorithme de [CHA 791 conçu pour un anneau unidirectionnel. Quant a l'alqorithme base sur une exploration en largeur il a une complexite plus élevée que celles des autres algorithmes a part celui l'algorithme de [HIR 811.
L'exploration en larqeur a une complexite qui tend a depasser celle de [HIR 801 au delà d'une taille de réseau egale a 40.

L' algorithme de [DOL 821 occupe la deuxieme place après les algorithmes de Chang [CHA 791 et l'exploration en profondeur. Les algorithmes de Korach et Korach modifié viennent en troisieme position.

Sur un anneau décroissant(figures IV.4a, IV.4b), l'exploration en largeur est mieux que l'algorithme de Chang [CHA 791. L'exploration en profondeur est la plus performante. Elle est suivie de l'algorithme de Dolev [DOL 821.

b) Comparaison des temps d'blection

Cette comparaison est representee par la figure IV.1c pour le mode asynchrone, et la figure IV.1d pour le mode synchrone. L'algorithme de [HIR 801 donne la plus mauvaise performance dans les deux modes d'execution, synchrone et asynchrone. L'exploration en largeur vient en deuxieme place. Quant aux algorithmes d'exploration en profondeur et les algorithmes de Chang, Dolev et Korach [CHA 79, DOL 82, KOR 81, Korach-21, ils occupent la premiere place avec des complexités presque égales surtout en mode synchrone.

1.2 Election individuelle

a) Comparaison des trafics

La figure IV.5a montre les elections individuelles en mode asynchrone. L'exploration en profondeur ainsi que les algorithmes de Chang [CHA 791, de Korach modifie ont des complexites qui ne different pas beaucoup. La version initiale de l'algorithme de Korach [KOR 811 vient en deuxieme place. L'exploration en largeur a une complexite moins bonne, tandis que l'algorithme de Hirshberg [HIR 801 presente la complexite la plus mauvaise. La figure IV.5b montre le cas synchrone et les mêmes observations peuvent être faites.

b) Comparaison des temps d'élection

En mode synchrone(figure IV.5d), l'algorithme de Korach modifie est le plus performant, suivi de l'algorithme d'exploration en profondeur et de celui de Chang[CHA 791. Ensuite l'algorithme de [KOR 81] et l'exploration en largeur sont situes en troisieme place. Enfin, et en derniere place c'est encore l'algorithme de [HIR 80] qui presente la plus mauvaise performance.

Pour le mode asynchrone(figure IV.5c), 1 ' algorithme de Korach modifie l'exploration en profondeur, et l'algorithme de chang occupent la premiere place. 11 vient ensuite l'algorithme de Korach [KOR 811 ainsi que l'exploration en largeur. L'algorithme [HIR 801 occupe la derniere place avec un retard net par rapport aux algorithmes precedents.

2 Topologie complete

Les alqorithmes compares pour cette topologie sont les algorithmes A, B et C de Y.Afek [AFE 851, l'algorithme de Loui [LOU 861, ainsi que les algorthmes de Helary [HEL 86bl.

2.1 Election collective

a) Comparaison des trafics

Les figures IV.8a et IV.8b indiquent clairement que l'exploration en profondeur a une complexite en messages moins élevée que celle des algorithmes A, B, C [AFE 851 et celui de [LOU 861 qui ont les mêmes complexites. L'exploration en largeur donne la complexite la plus élevée.

b) Comparaison des temps d'élection

En mode synchrone(figure IV.8d), ce sont les algorithmes de Loui [LOU 861 et l'exploration en profondeur qui offrent le meilleur temps. En deuxi &me place vient l'exploration en largeur puis l'algorithme A. Les algorithmes B et C sont les plus lents.
En mode asynchrone(figure IV.8c), c'est l'alqorithme de Loui qui

est le plus performant. L'exploration en profondeur vient en deuxieme place. L'exploration en largeur occupe la troisieme position et la quatrieme revient a l'algorithme A [AFE 851 suivi directement de l'algorithme C, puis B.

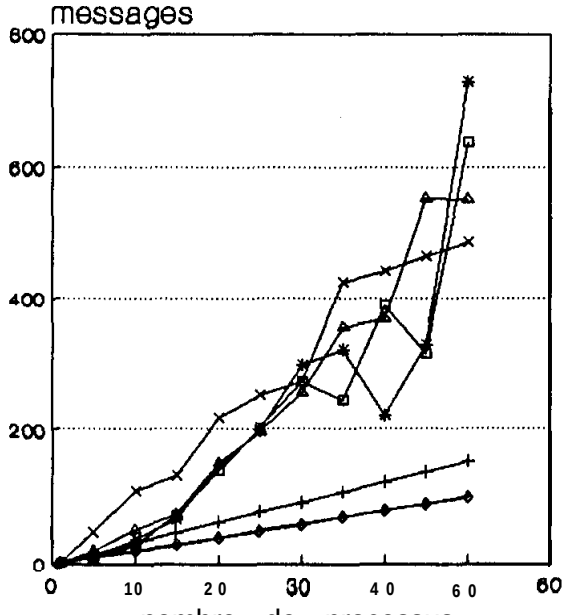
2.2 Election indivlduelle

a) Comparafson des trafics

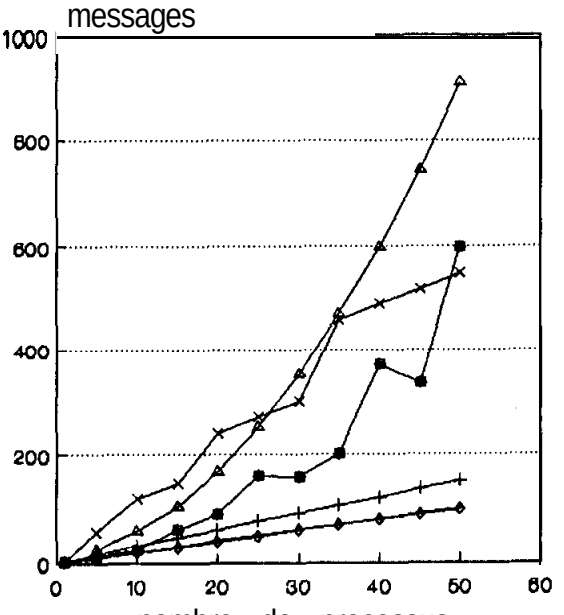
Sur les figures IV.10a, IV.lOb, il apparait que l'exploration en profondeur est la plus performante en ce qui concerne la complexite en messages. Après cet algorithme, on trouve les trois algorithmes proposes dans [AFE 85], A, B, C. L'algorithme de [LOU 861 est classé en troisieme place avant l'exploration en largeur qui donne la plus mauvaise complexite.

b) Comparaison des temps d'élection

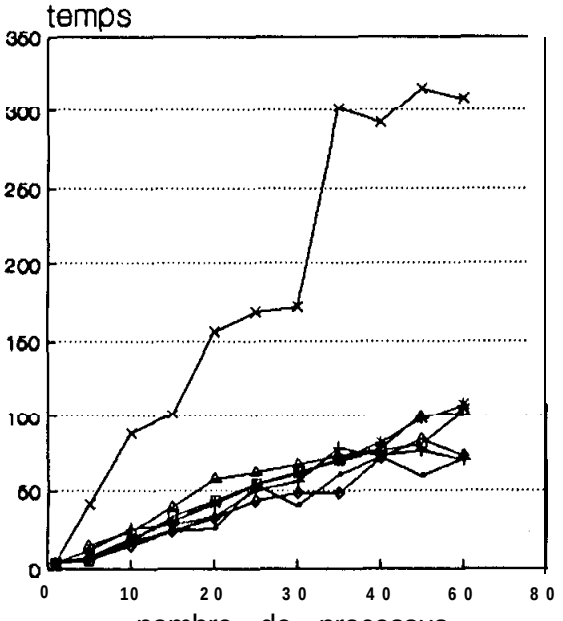
En tenant compte des deux diagrammes(figures IV.10c, IV.10d) il est possible de conclure que l'exploration en profondeur est la plus rapide. En deuxi ème place vient l'algorithme de Loui suivi de l'exploration en largeur. La derniere place revient aux trois algorithmes A, B, C.



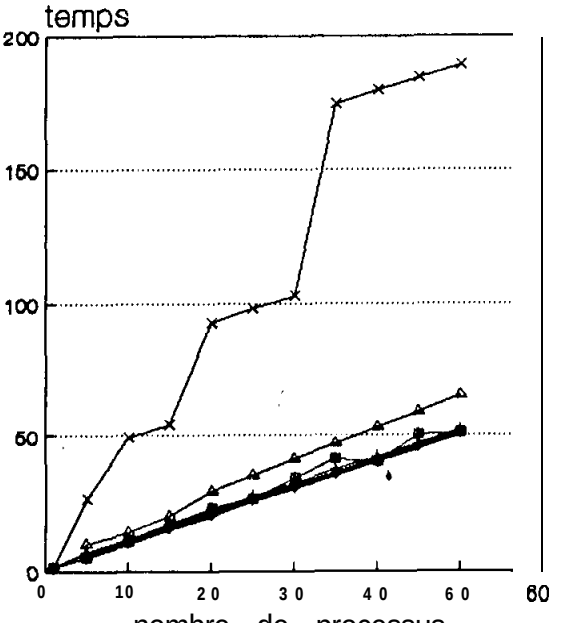
Flgure 1.a: Asyncrone



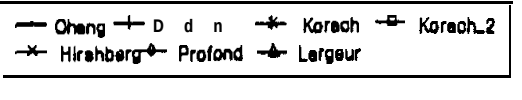
Flgure 1.b: Synchrone



Flgure 1.c: Asynchrone



Flgure 1.d: Synchrone



Flgure IV.1: Collective (croissant)

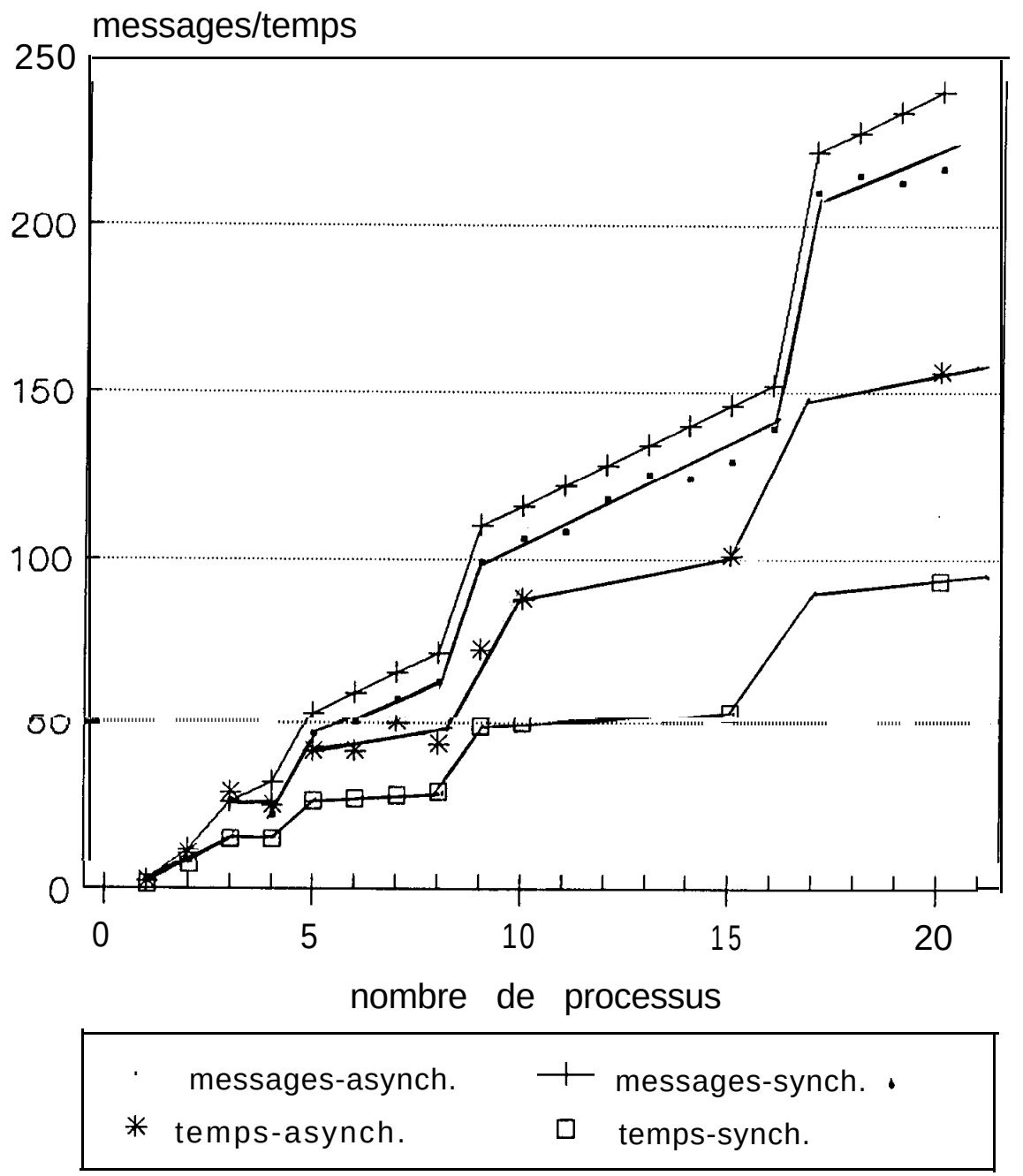


Figure 1.e: Algorithme de Hirshberg

Figure IV.1: Collective (croissant)

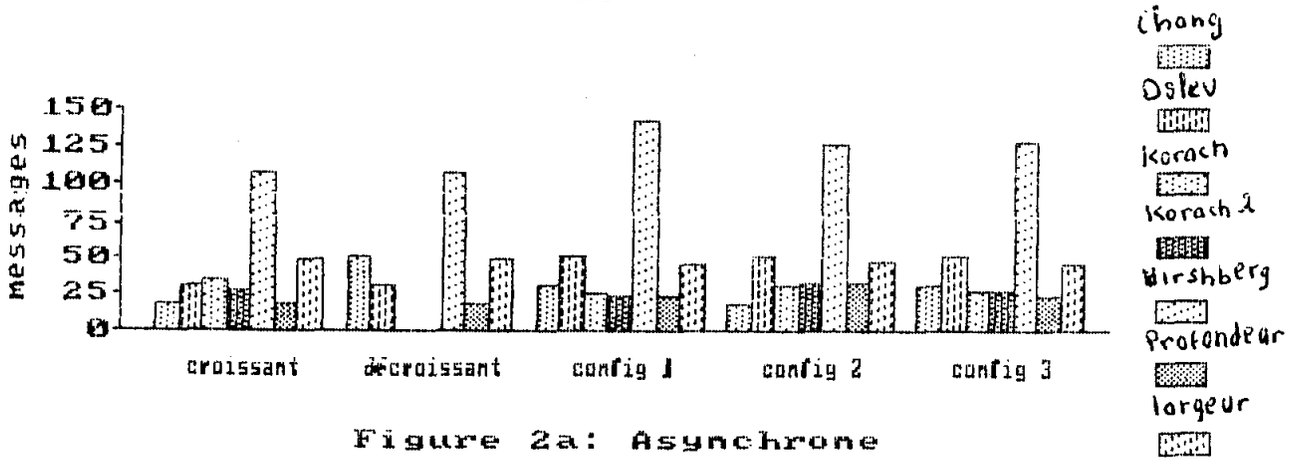


Figure 2a: Asynchrone

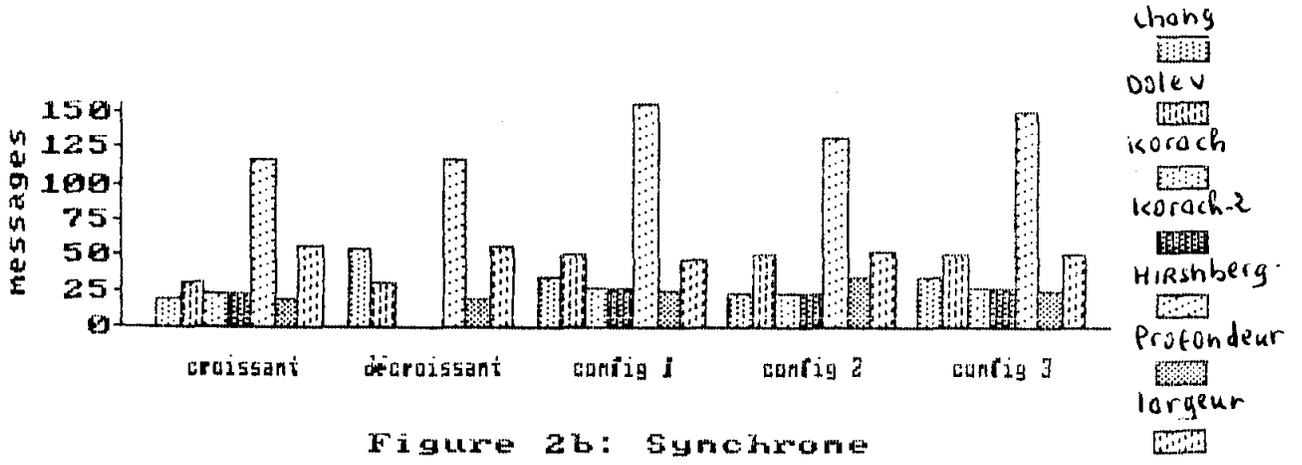


Figure 2b: Synchrone

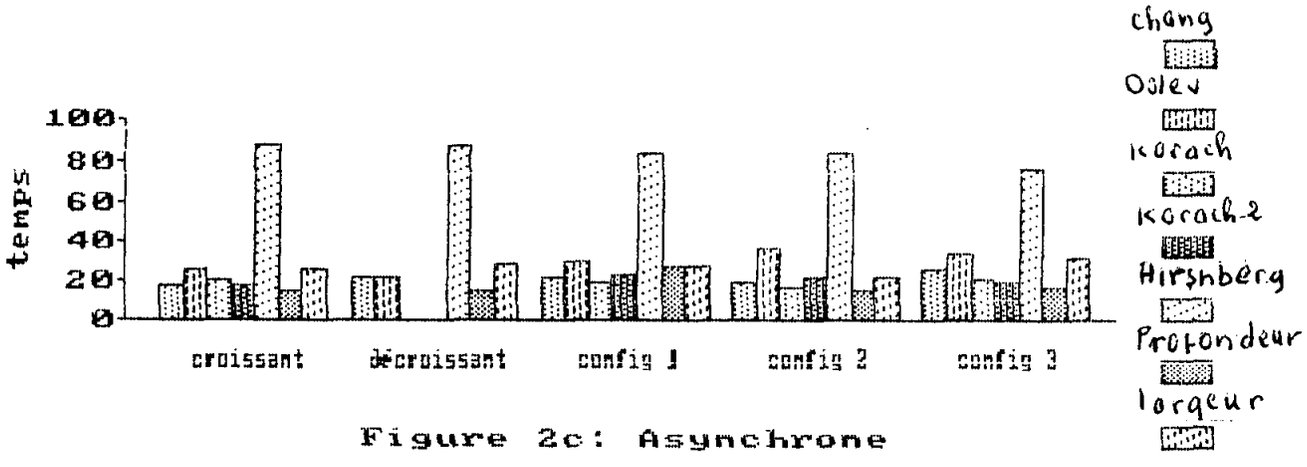


Figure 2c: Asynchrone

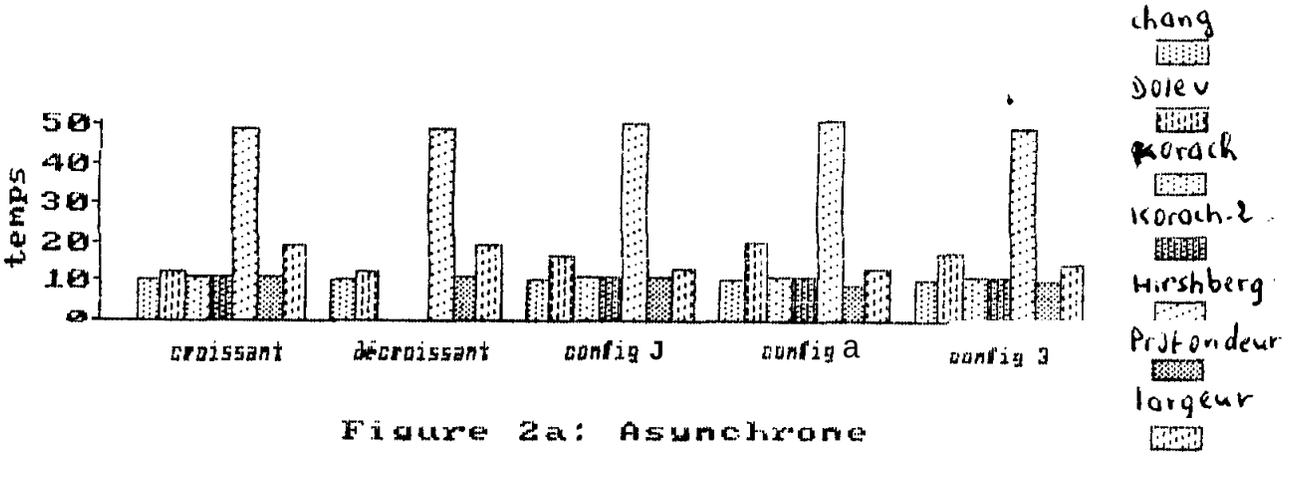


Figure 2a: Asynchrone

Figure IV.2: Collective (anneau)

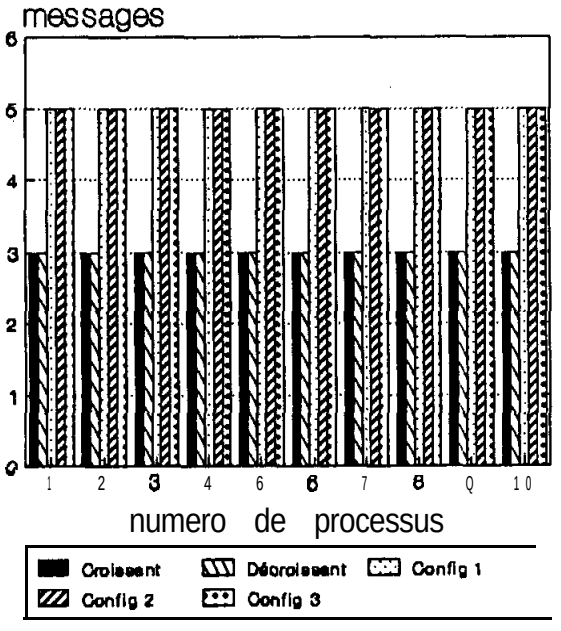


Figure 3a: Asynchrone et Synchrone

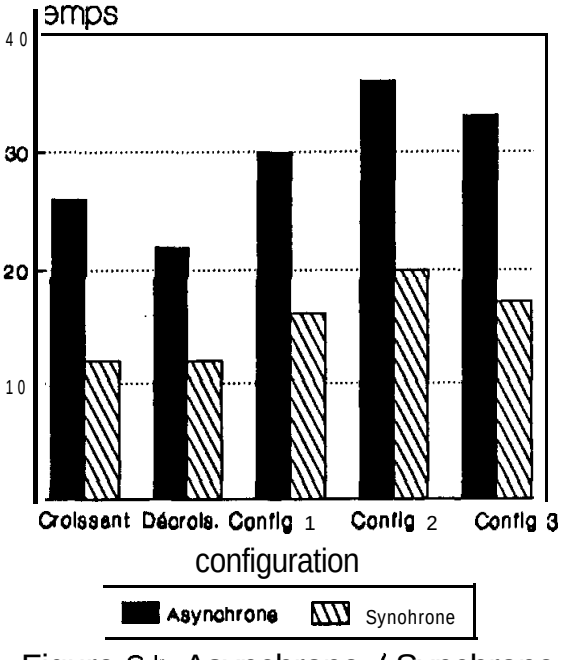


Figure 3.b: Asynchrone / Synchrone

Figure IV.3: Algorithme de Dolev

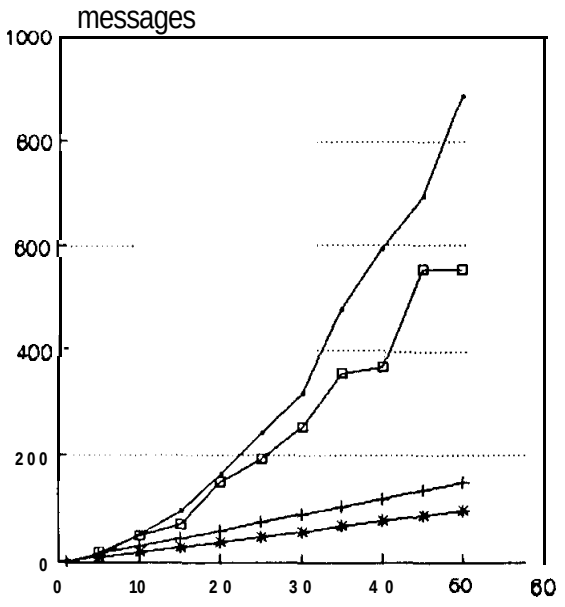


Figure 4.a: Asynchrone

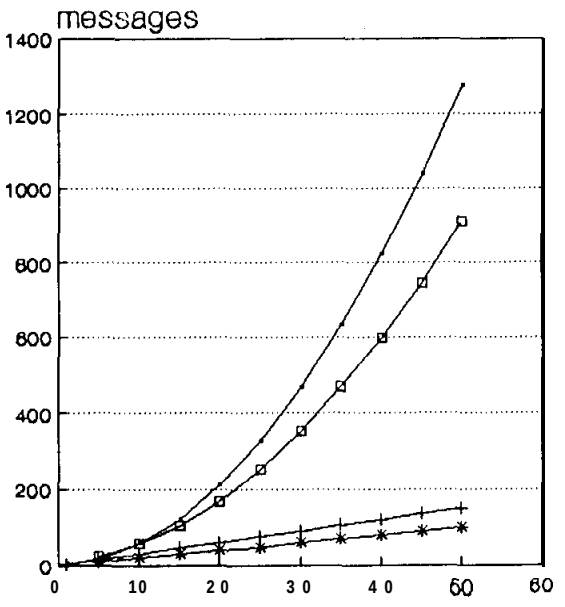


Figure 4.b: Synchrone

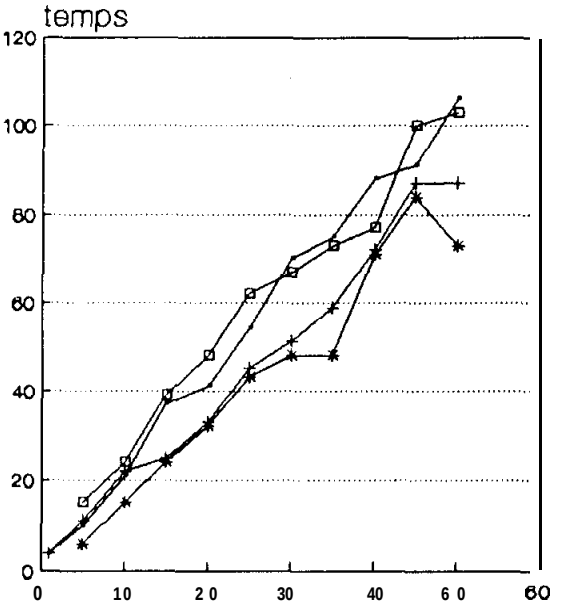


Figure 4.c: Asynchrone

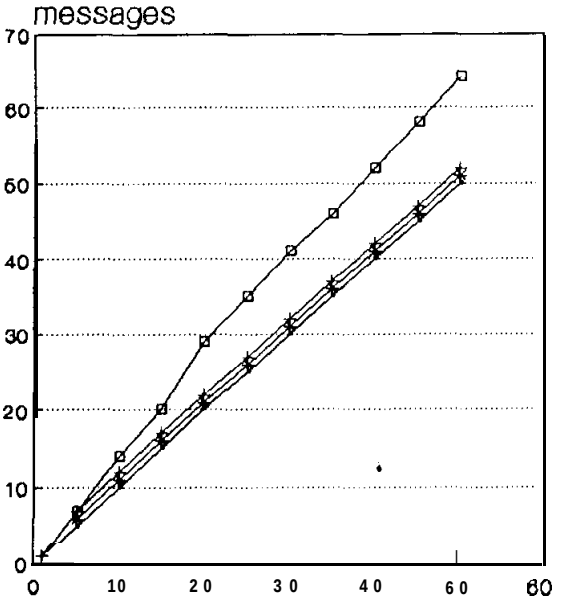


Figure 4.d: Synchrone

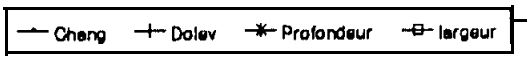


Figure IV.4: Collective (&croissant)

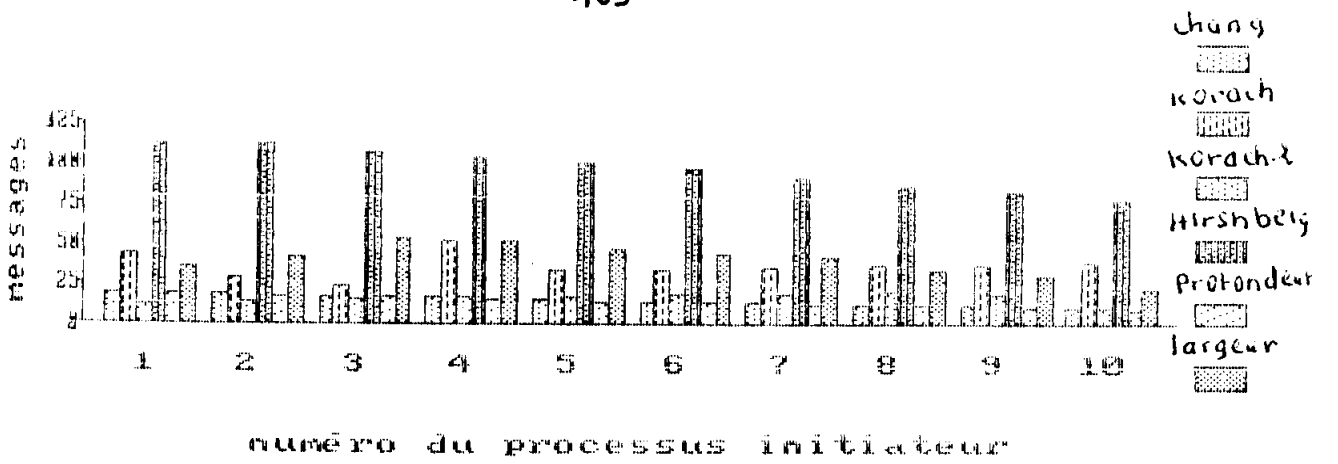


Figure 5a: Asynchrone

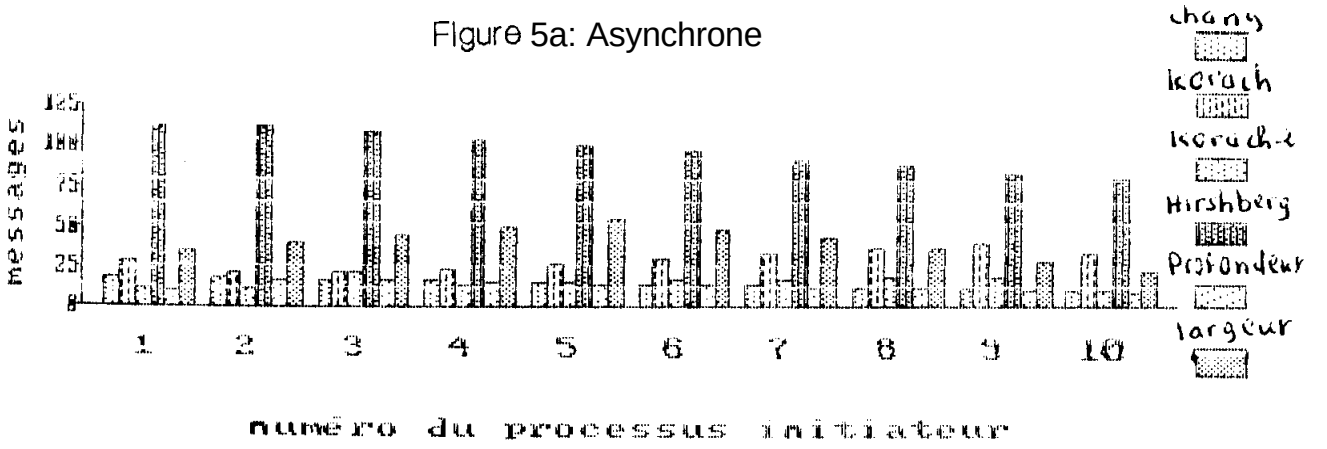


Figure 5b: Synchrone

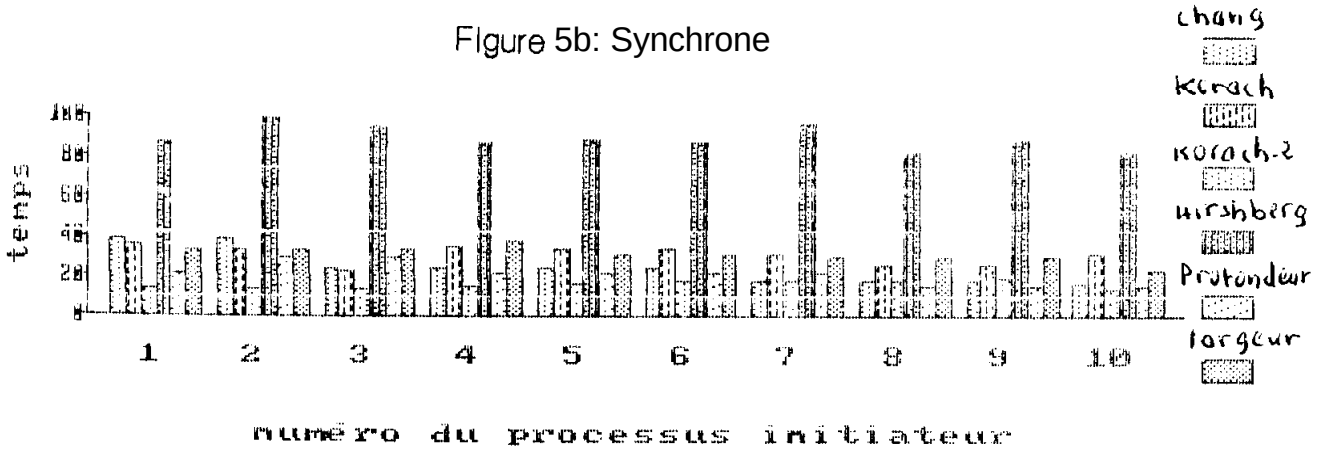


Figure 5c: Asynchrone

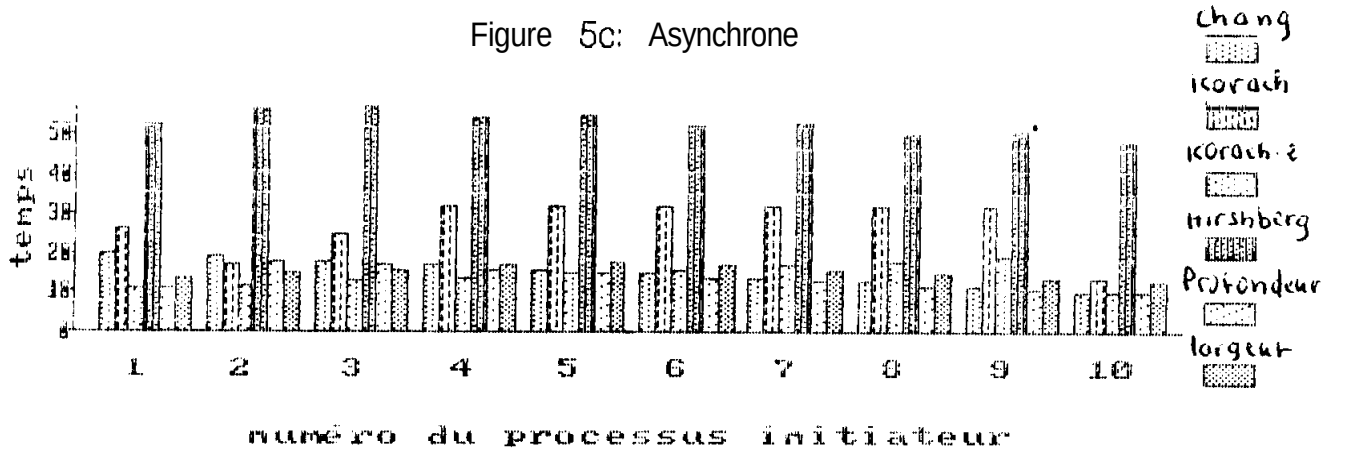


Figure 5d: Synchrone

Figure IV.5 Individuelle

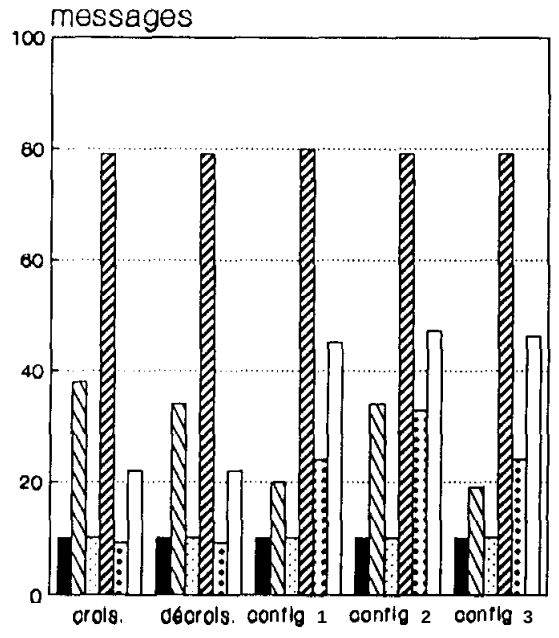


Figure 6.a: Asynchrone

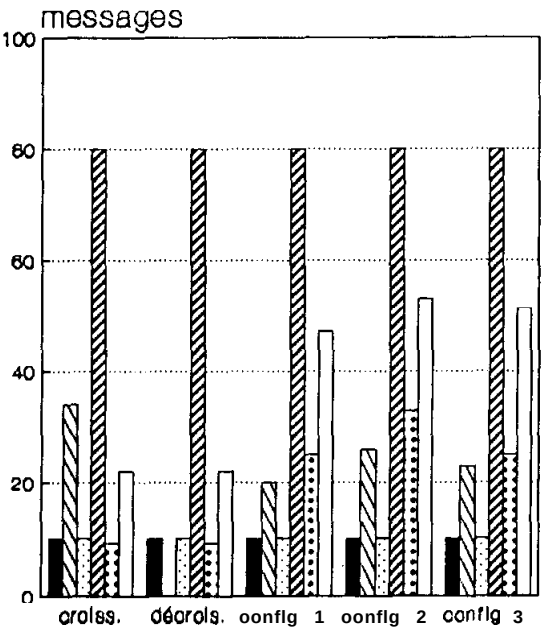


Figure 6.b: Synchrone

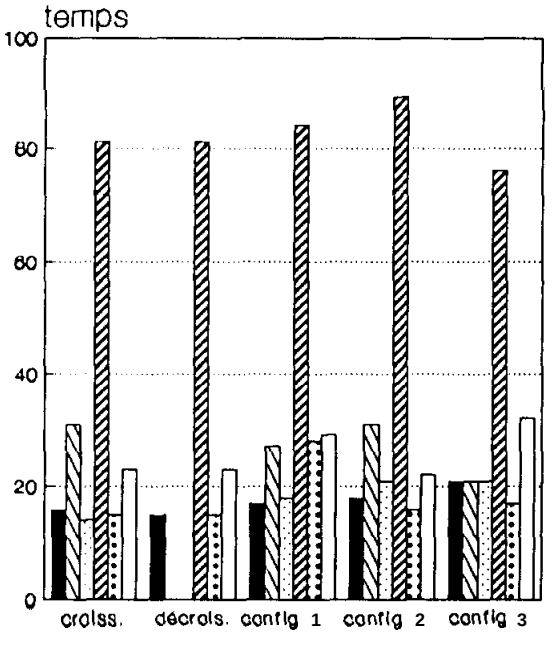


Figure 6.c: Asynchrone

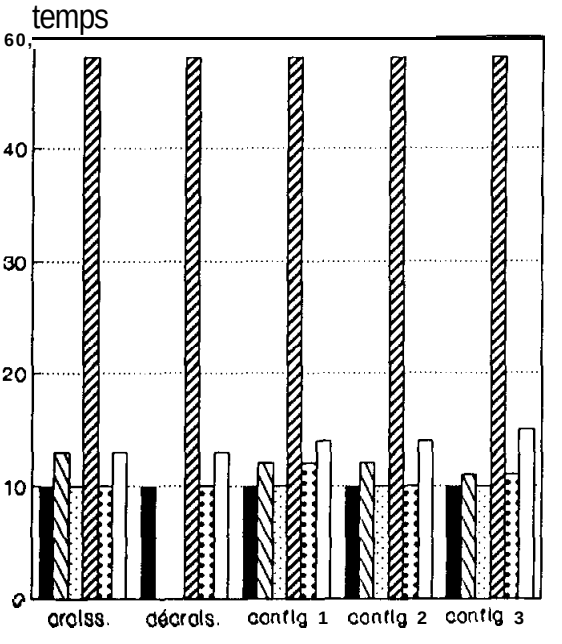
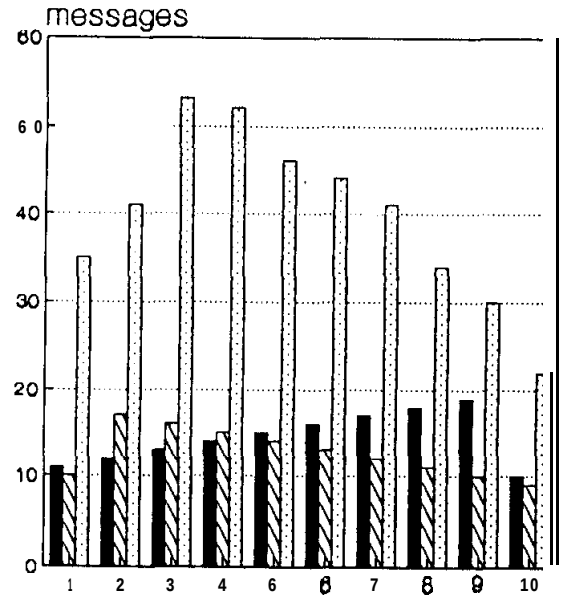
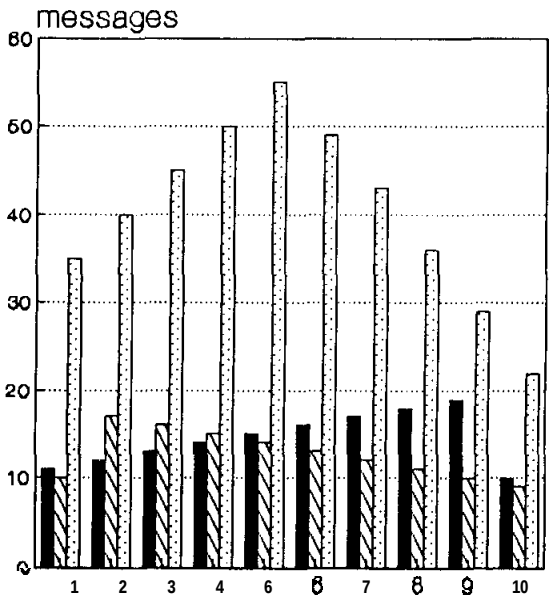


Figure 6.d: Synchrone

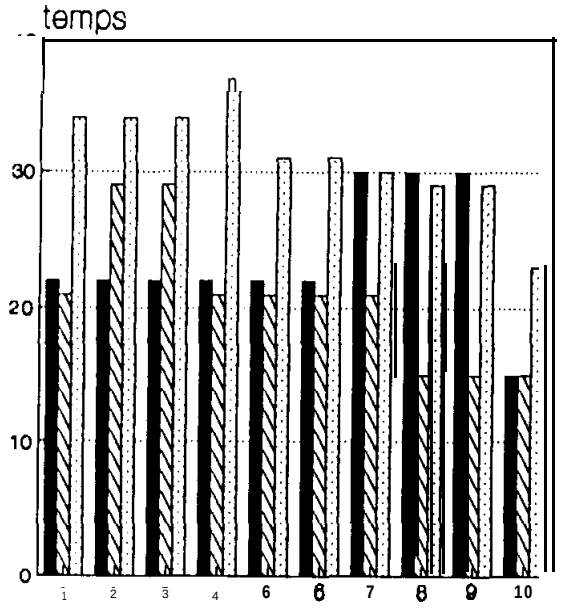
Figure IV.6: Individuelle (Pmax Mt.)



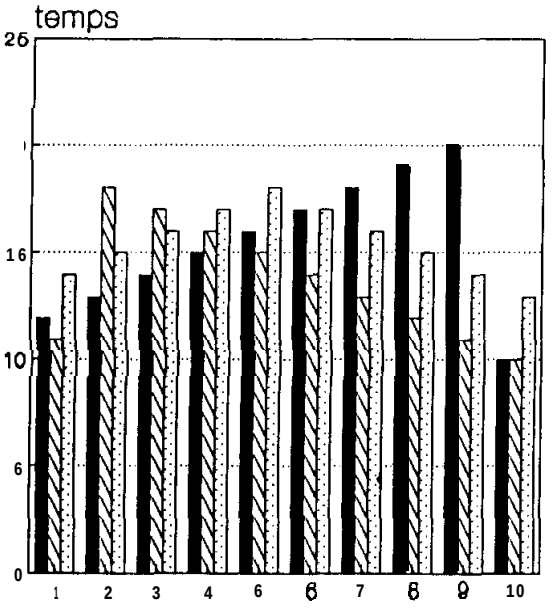
numéro du processus Initiateur
Figure 7.a: Asynchrone



numéro du processus Initiateur
Figure 7.b: Synchrone



numéro du processus Initiateur
Figure 7.c: Asynchrone



numéro du processus Initiateur
Figure 7.d: Synchrone

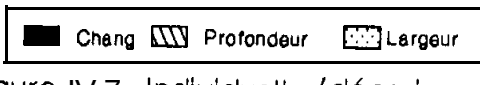


Figure IV.7: Individuelle (décroissant)

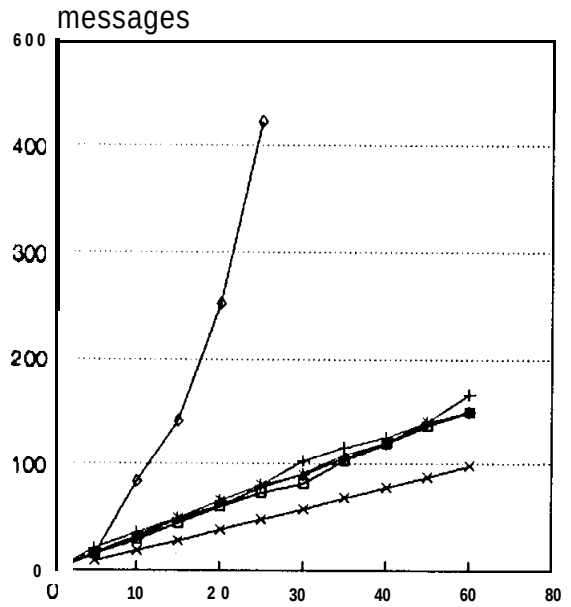
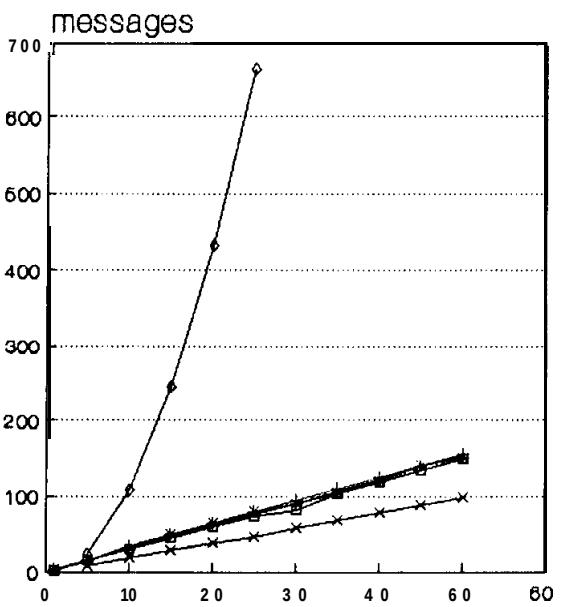
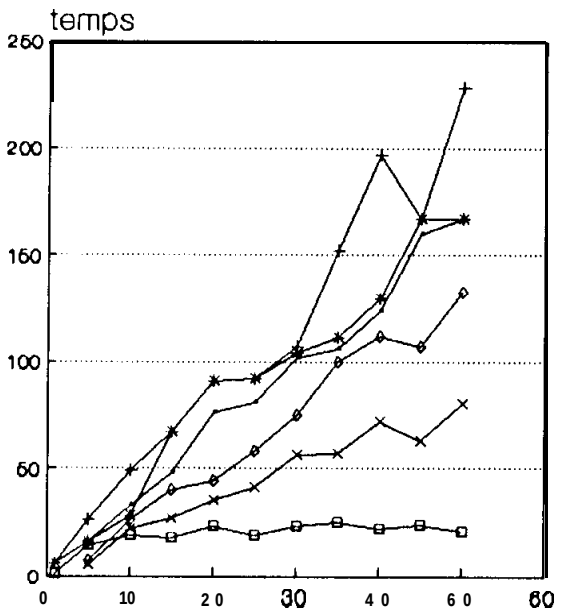


Figure 8.a: Asynchrone



F lgure 8. b: Synchrone



Flgure 8.c: Asynchrone

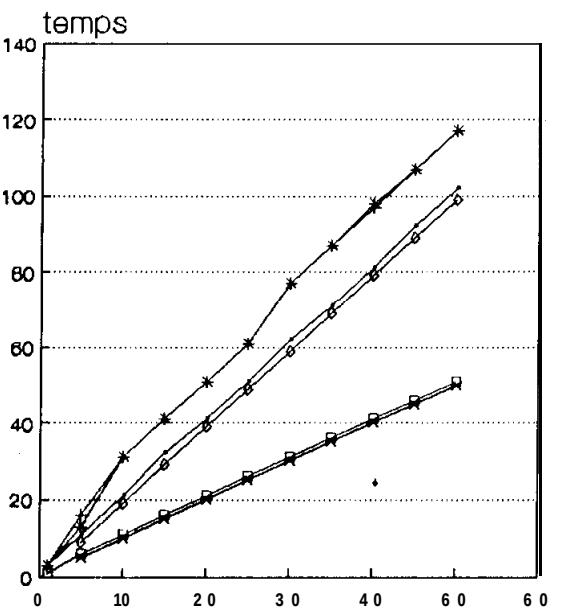


Figure 8.d: Synchrone

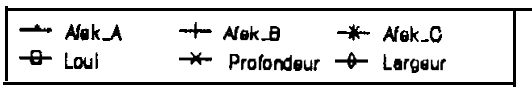


Figure IV.8: Collective (complet)

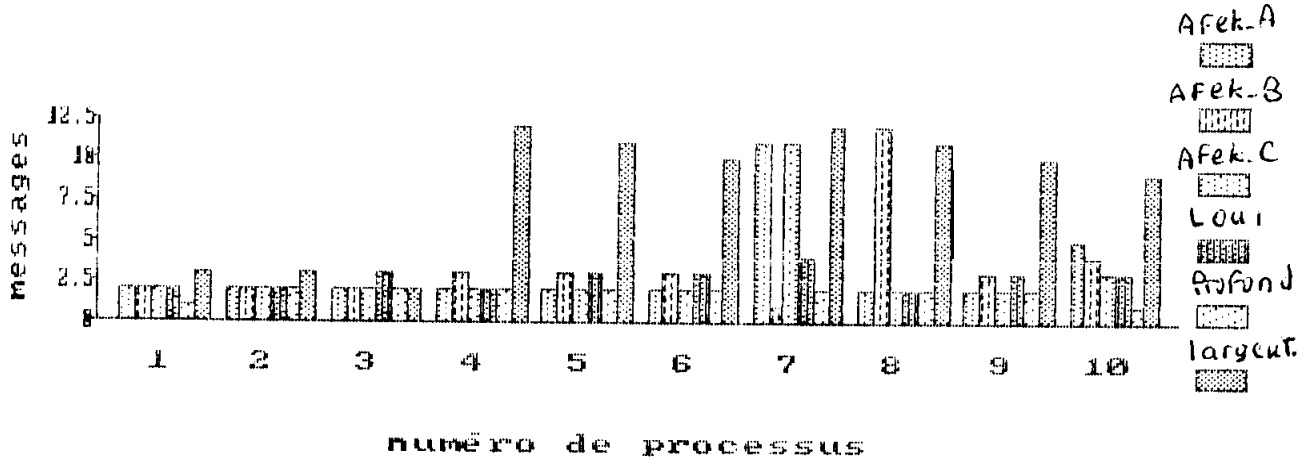


Figure 9a: Asynchrone

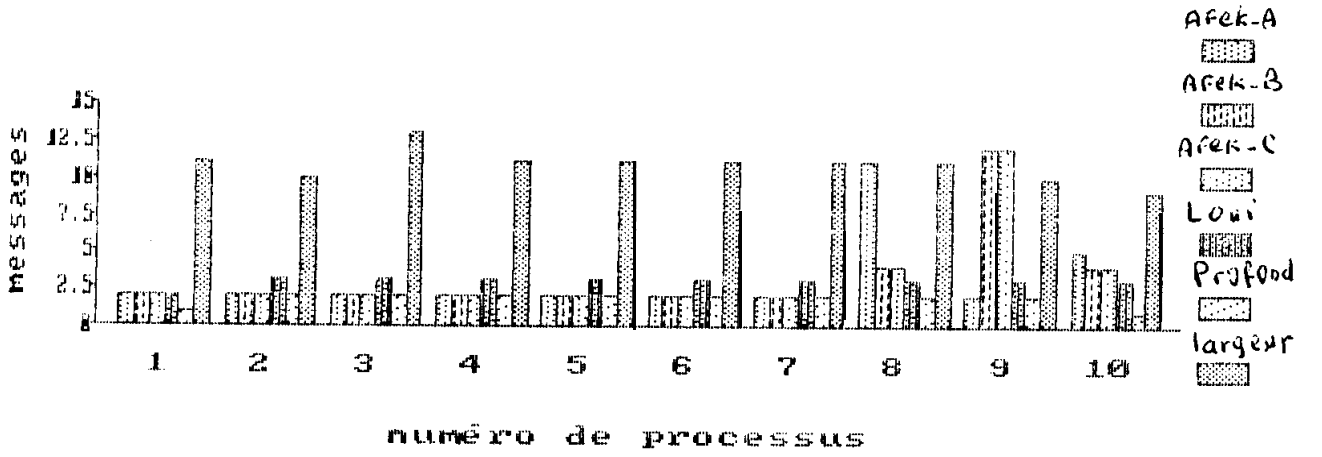


Figure 9b: Synchrone

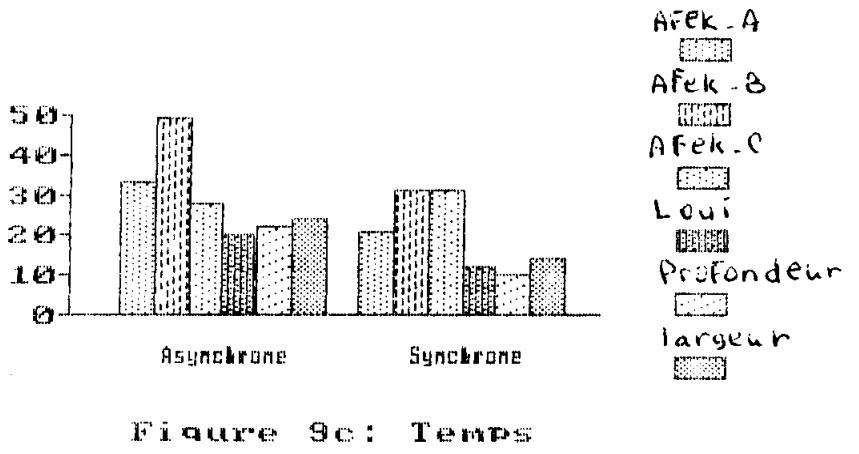


Figure IV.9: Collective (complet)

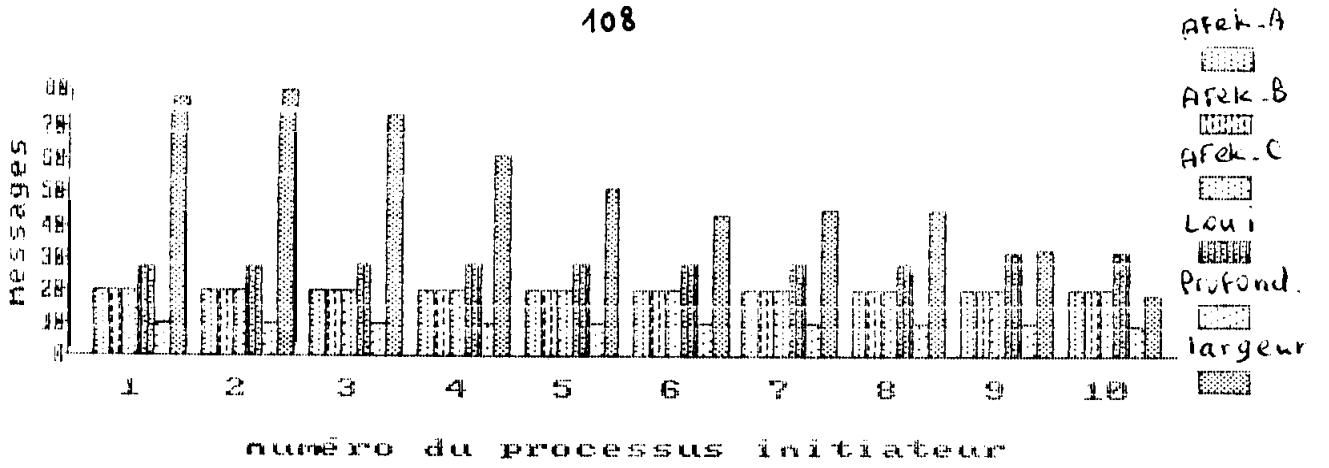


Figure 10a: Asynchrone

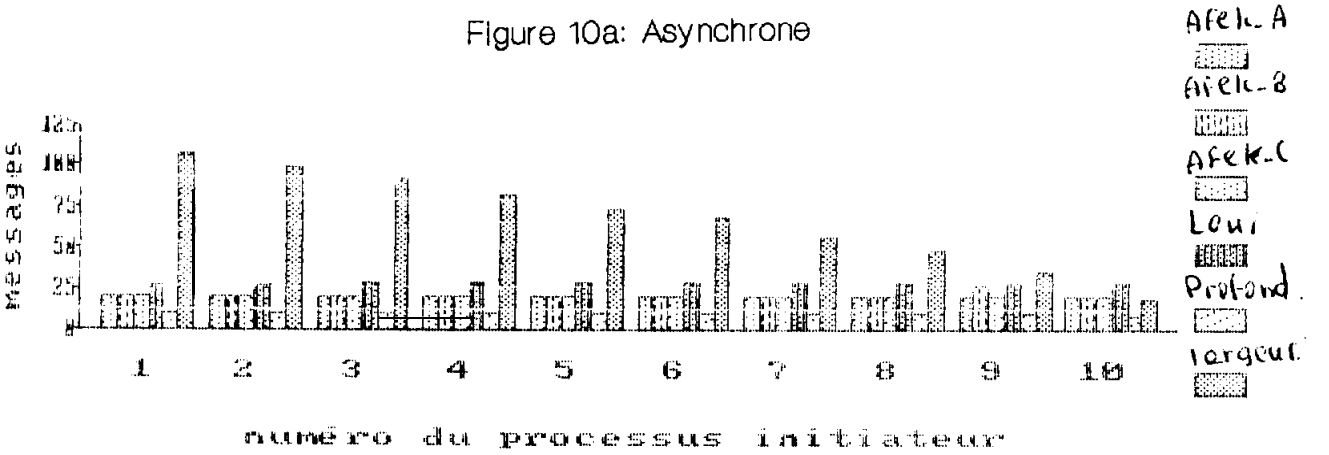


Figure 10b: Synchrone

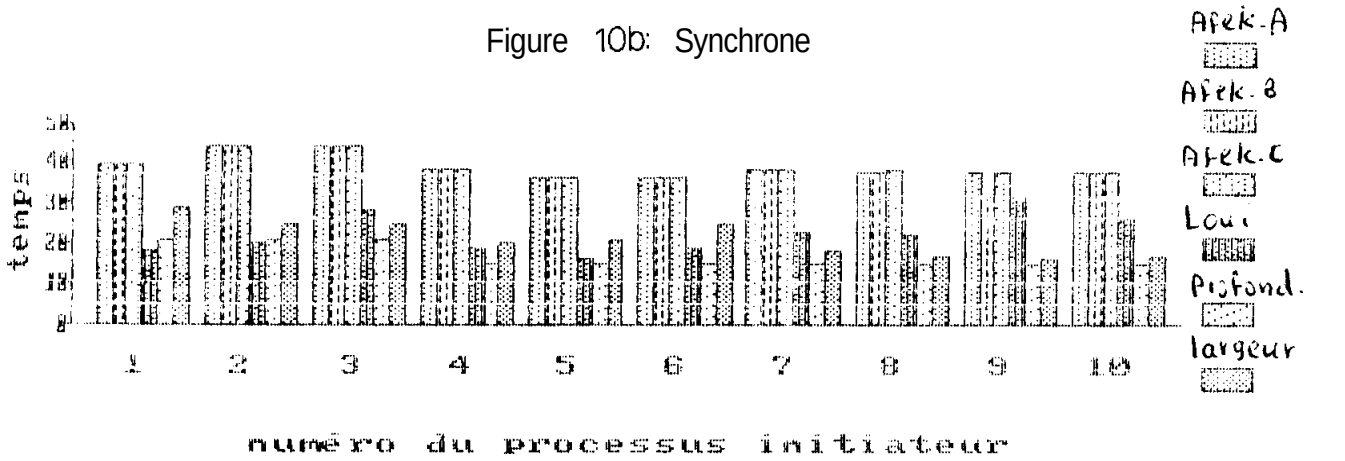


Figure 10c: Asynchrone

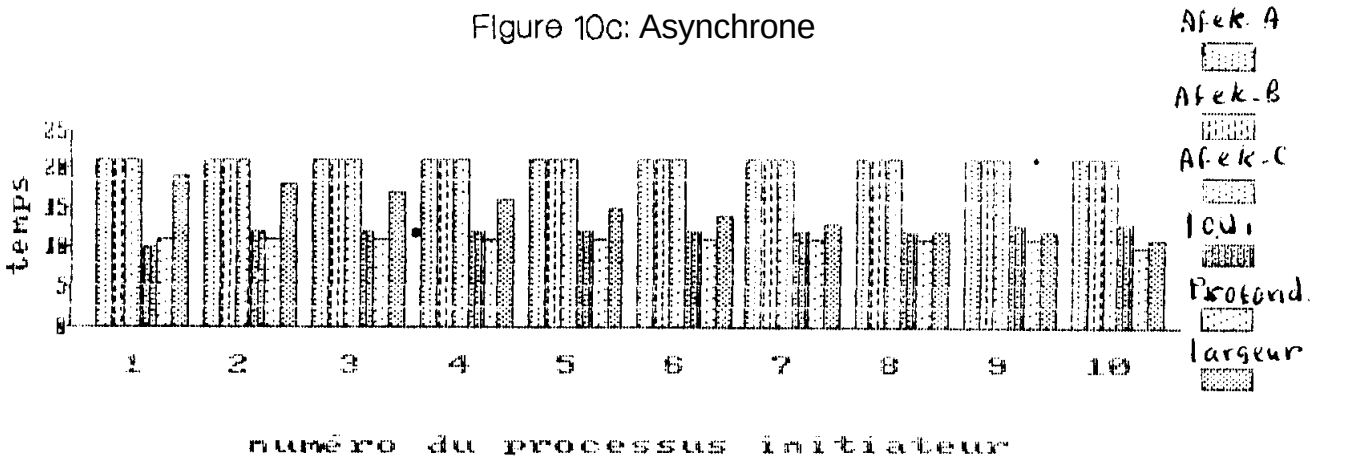


Figure 10d: Synchrone

Figure IV.10: Individuelle

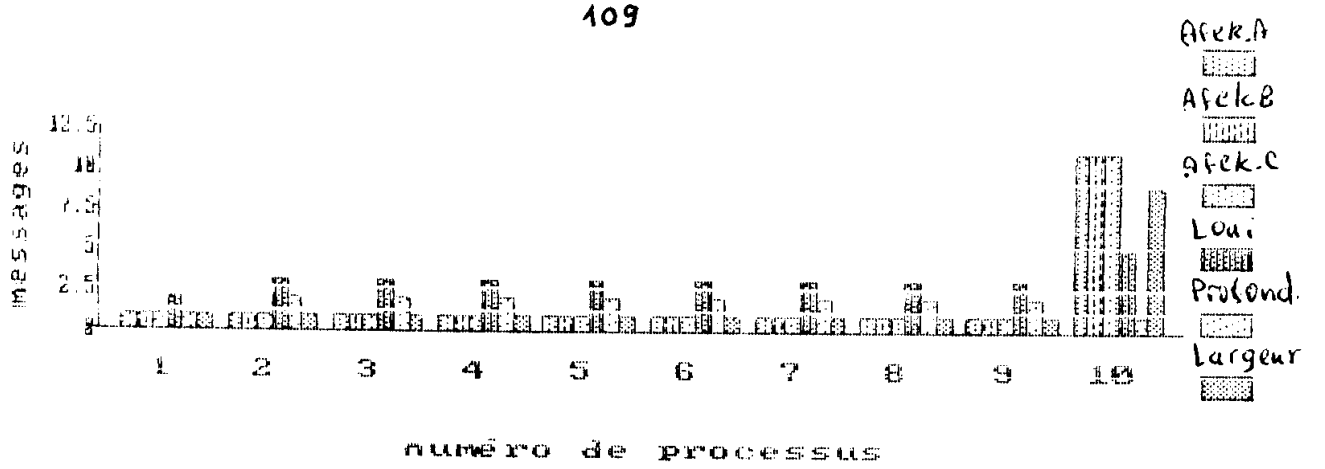


Figure 11a: Asynchrone

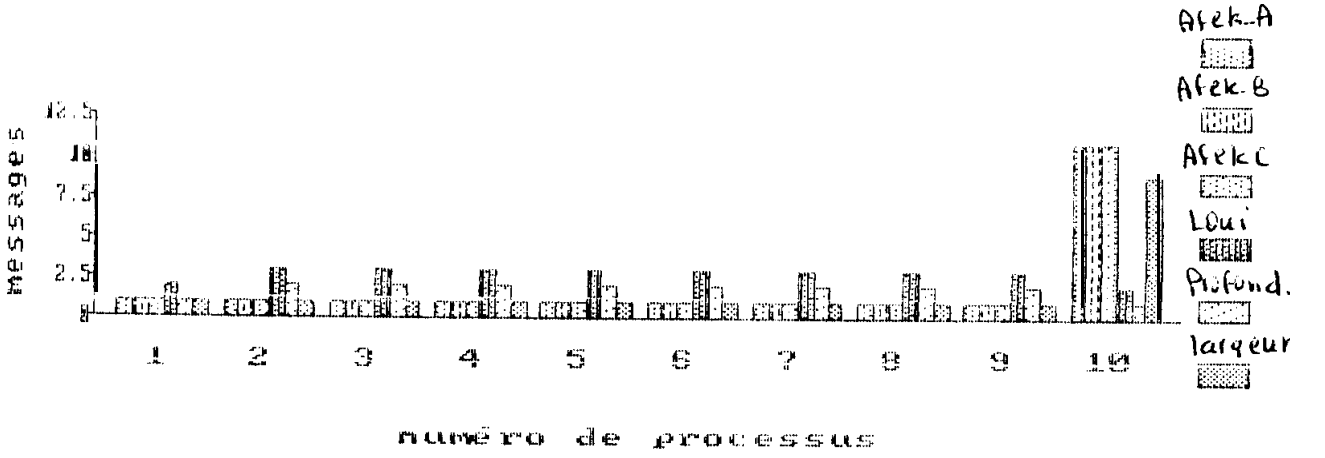


Figure 11b: Synchrone

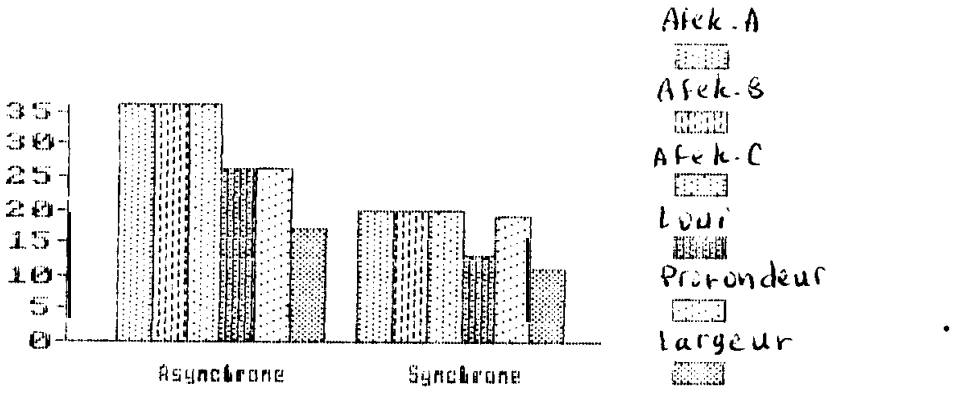
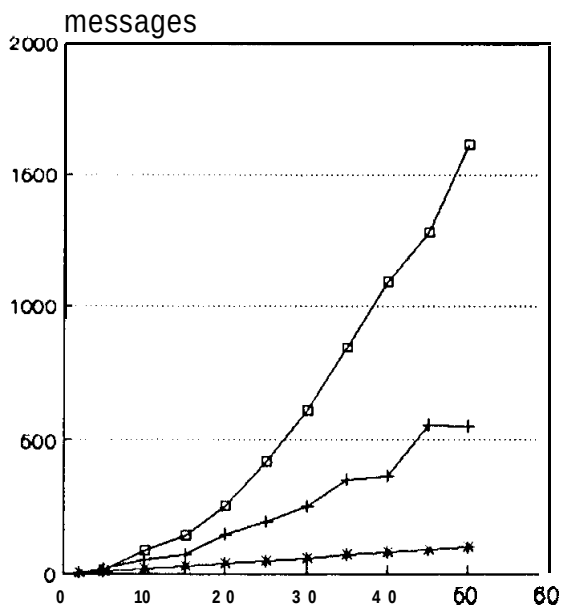


Figure 11c: Temps

Figure IV.11: Individuelle (Pmax init.)



Flgure 12.a: Asynchrone

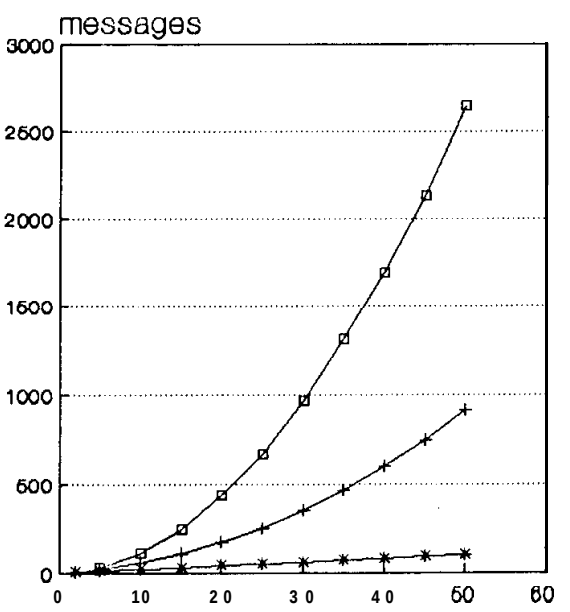
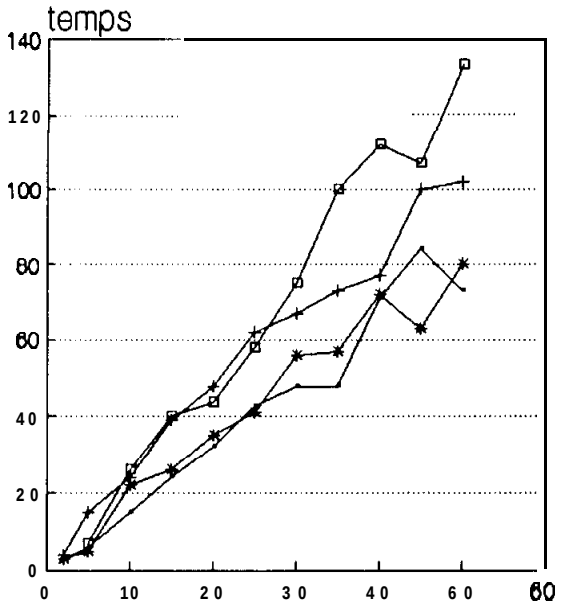
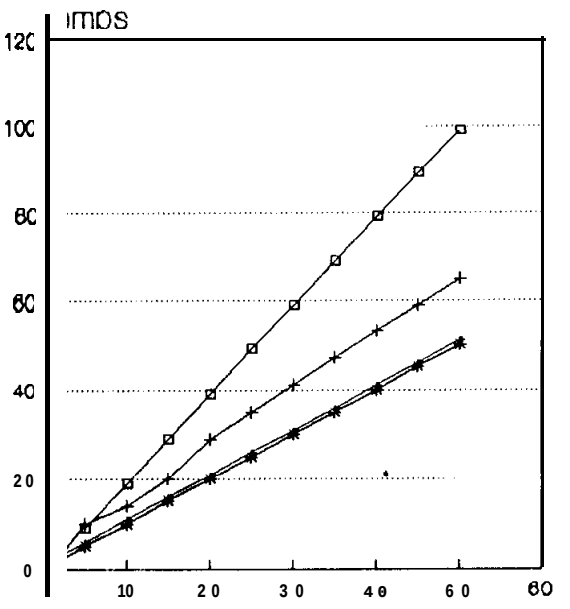


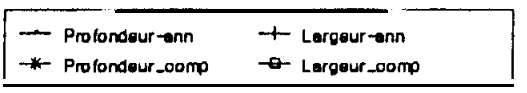
Figure 12.b: Synchrone



Flgure 12.c: Asynchrone



Flgure 12.d: Synchrone



Flgure IV.12: Collective

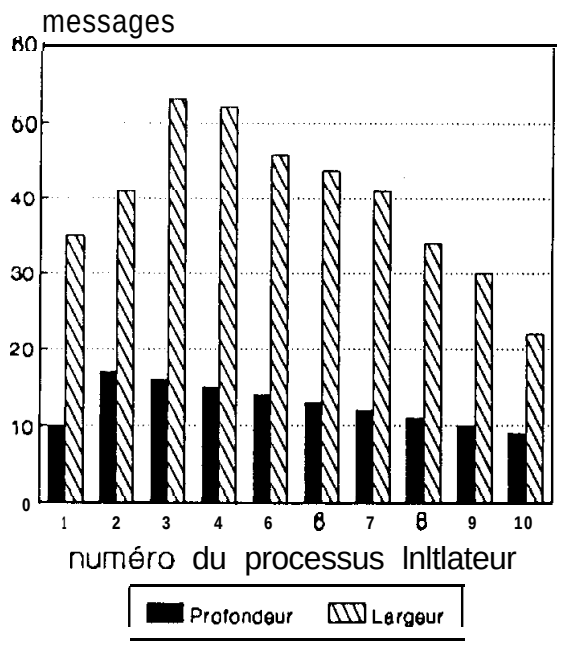


Figure 13.a: Asynchrone

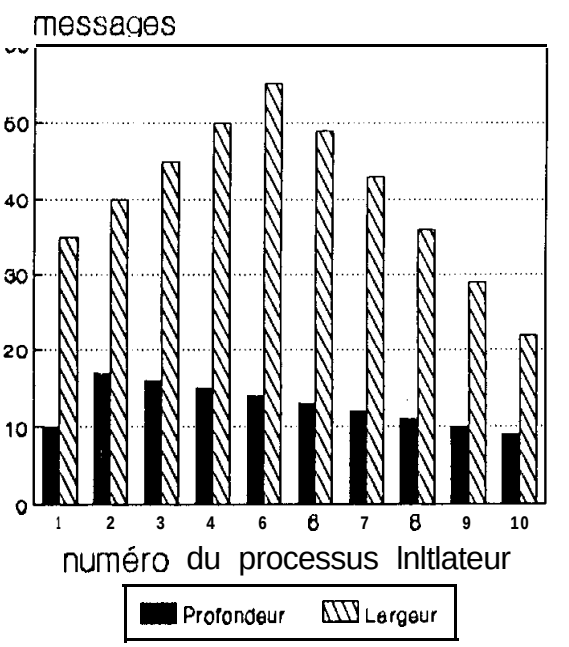


Figure 13.b: Synchrone

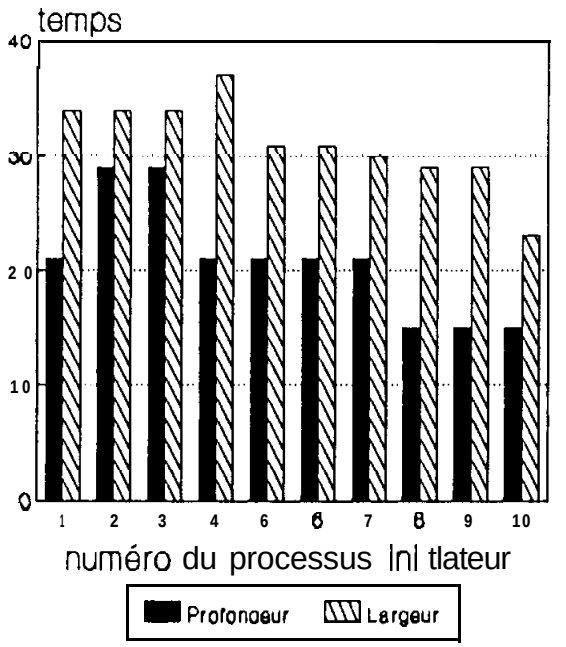


Figure 13.c: Asynchrone

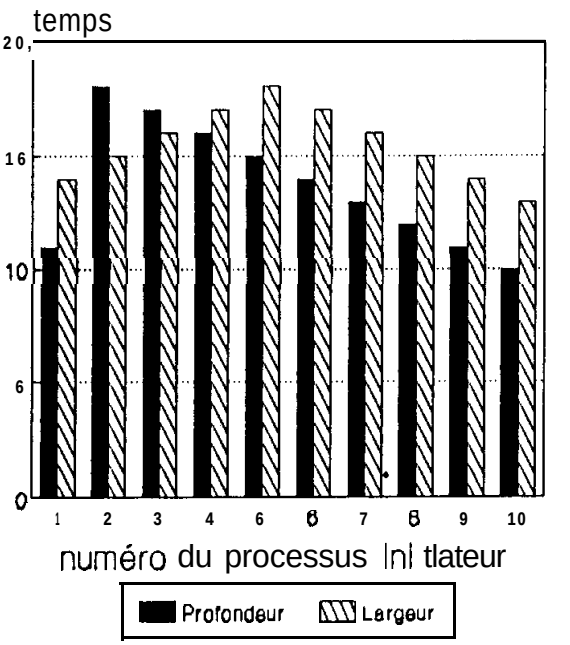


Figure 13.d: Synchrone

Figure IV.13: Individuelle (anneau)

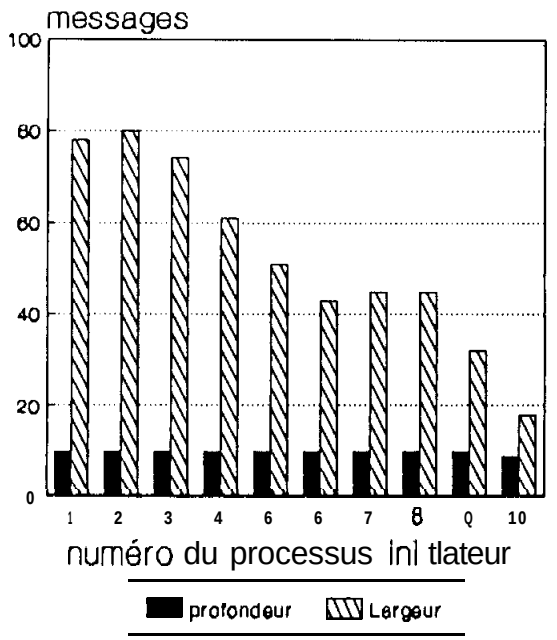


Figure 14.a: Asynchrone

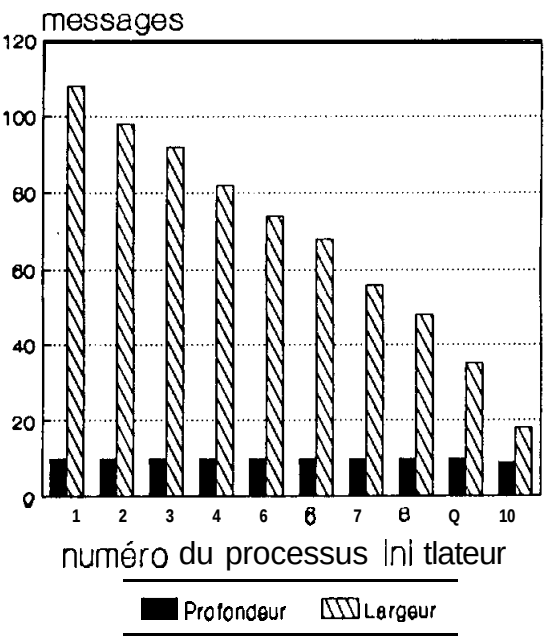


Figure 14.b: Synchrone

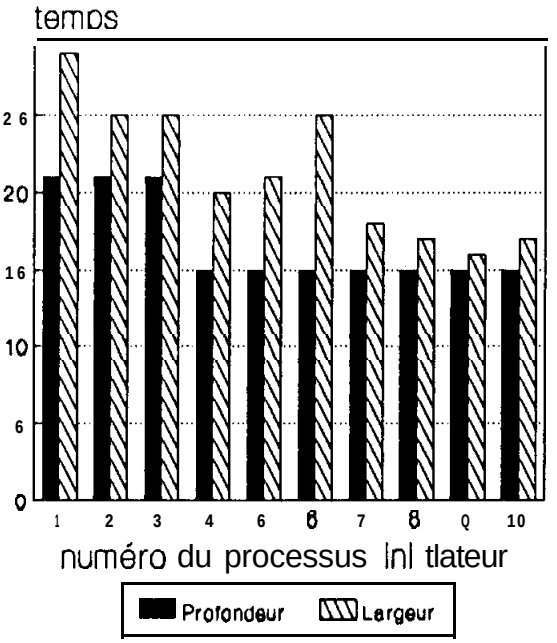


Figure 14.c: Asynchrone

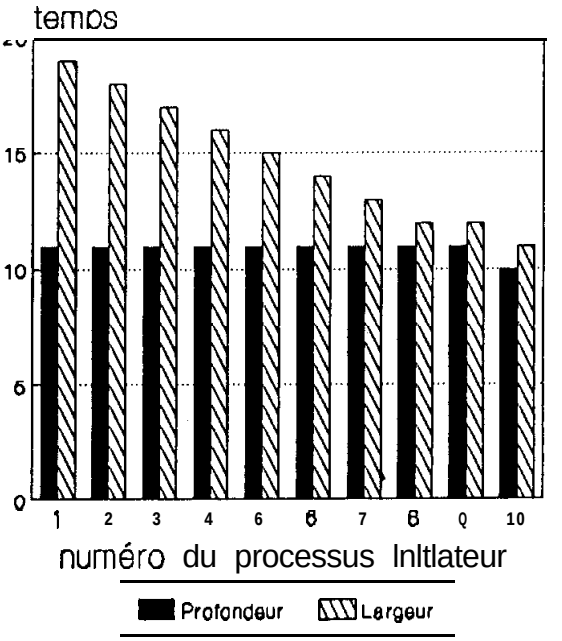


Figure 14.d: Synchrone

Figure IV.14: Individuelle (cornplet)

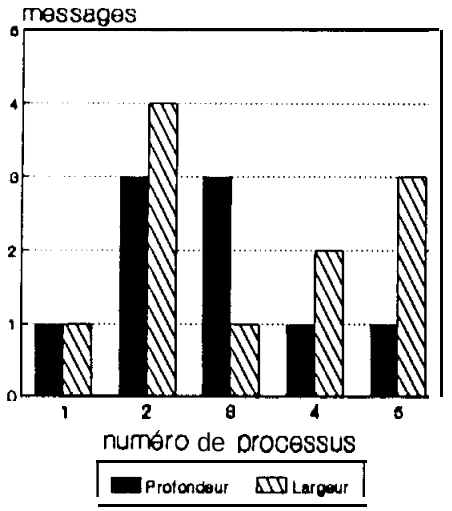


Figure 15.a: Asynchrone

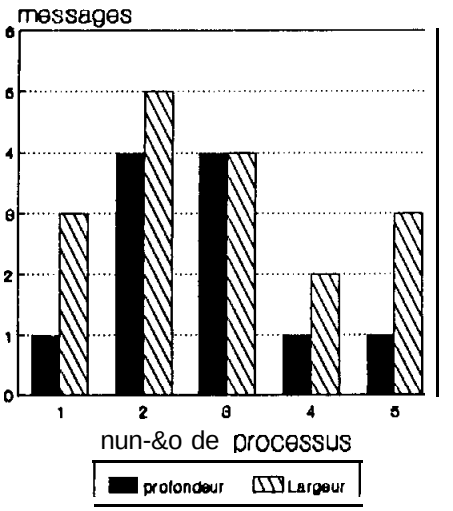


Figure 15.b: Synchrone

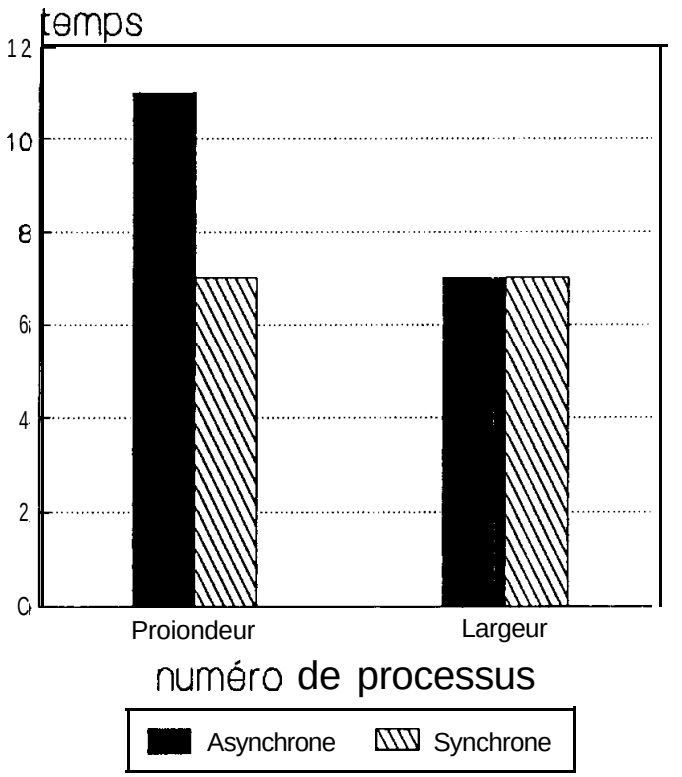


Figure 15.c: Asynchrone / Synchrone

Figure IV.15: Collective

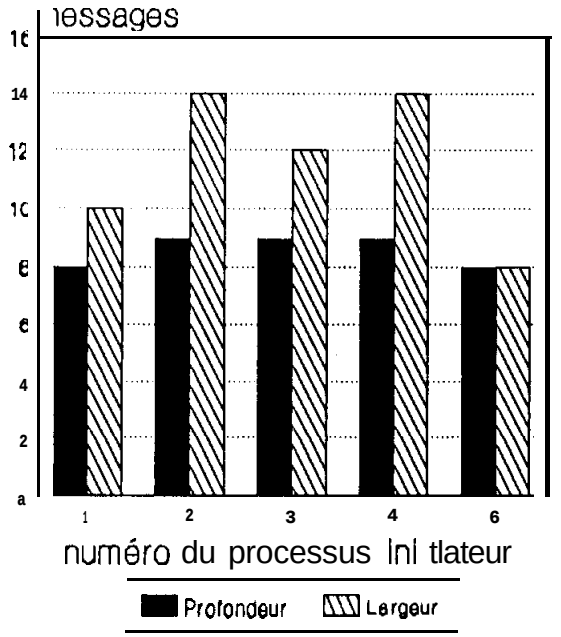


Figure 16.a: Asynchrone

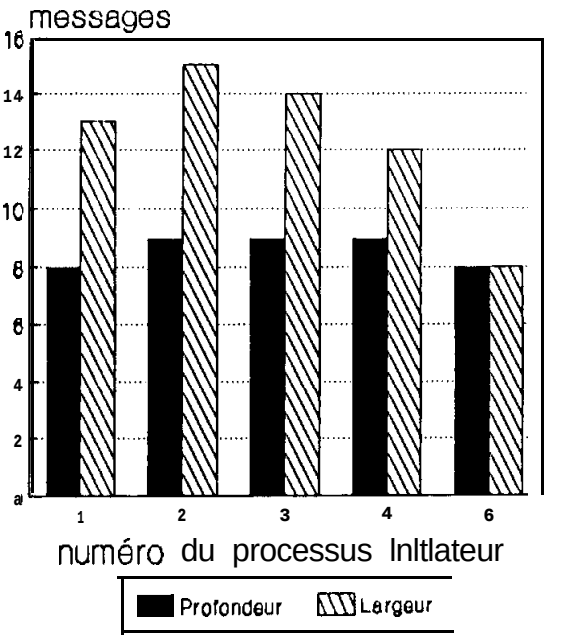


Figure 16.b: Synchrone

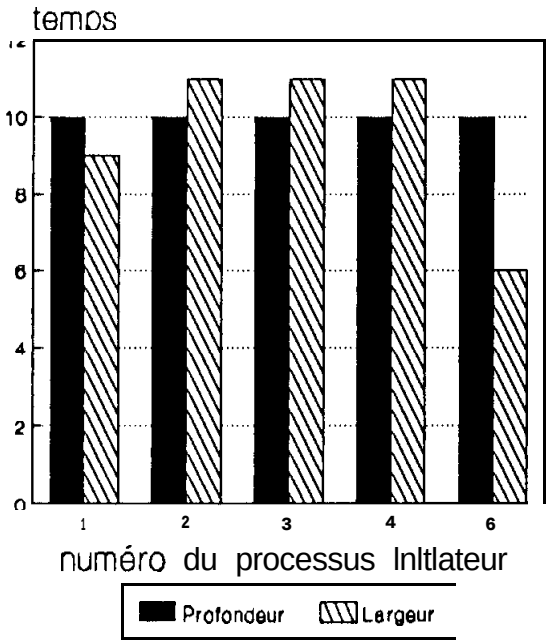


Figure 16.c: Asynchrone

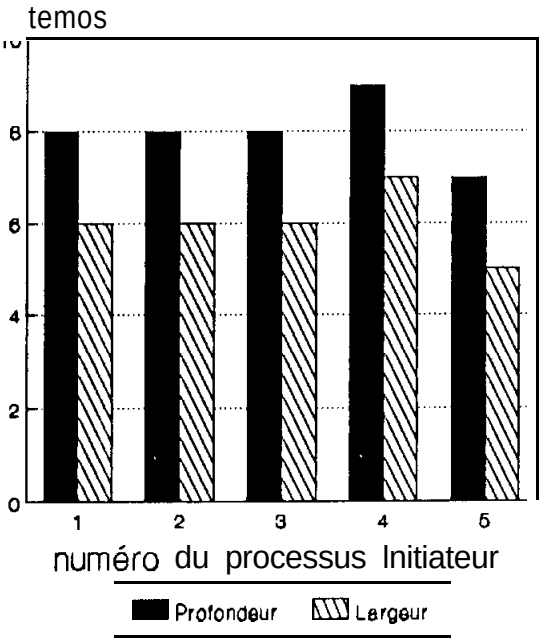


Figure 16.d: Synchrone

Figure IV.16: Individuelle

CONCLUSION

L'etude et l'evaluation des algorithmes distribues d'election nous ont permis de mieux saisir le comportement de ce type d'algorithmes. Concernant les resultats obtenus, outre les performances évaluées, et l'etude comparative de celles-ci, un certains nombre de resultats ont été obtenus:

- Detection d' erreurs. Par exemple, 1' algorithme de Loui [LOU 861 qui ne fonctionne pas sur certaines configurations de réseau(chapitre III).

- Confirmation de certaines complexites theoriques. Par exemple, le cas de l'alqorithme de chang [CHA 79].

- Obtention de complexites tout a fait differentes des complexites prevues. Par exemple, les algorithmes de Afek [AFE 851, et de Hirshberq [HIR 801.

- Certains comportements sont differents des previsions theoriques. Afek [AFE 851 a propose trois alqorithmes A, B et C qui ont été conçus de façon a ce que l'alqorithme B puisse ameliorer la complexité en messages de A. Ce dernier, ayant une complexité temporelle de O(n). L'alqorithme C, Ctait suppose réduire la complexité temporelle de B, en conservant sa complexité en messages. Or, l'evaluation a montre que les trois alqorithmes ont pratiquement les memes comportementslchapitre IV).

L'etude des algorithmes a permis de comparer leurs performances en nombre de messages et en temps. Le fait que cette evaluation ne depend pas d'un environement d'execution reel particulier permet d'affirmer que certains des résultats obtenus, tels que les resultats de la comparaison pourraient être généralisés a d'autres environements.

Les meilleurs algorithmes sont:

1- Pour la topoloqie en anneau:

-- anneau croissant:

collective		individuelle	
messages	temps	messages	temps
Chang Profondeur (+)	Chang Korach Korach-2	Profondeur	1. Korach-2 2. Profondeur 3. Chang (++)

(++): La numerotation siqnifie que les algorithme sont classes dans cet ordre, mais la difference est minime.

(+) : Ces alqorithme offrent pratiquement les mêmes performances.

-- anneau decroissant:

collective			individuelle	
messages	temps		messages	temps
	Async	Sync		
Profondeur Dolev	1.Profond 2.Dolev	Chang Profond	Profondeur	Profondeur

2- Pour la topologie à maillage complet:

collective			individuelle	
messages	temps		messages	temps
	Asyncl	Sync		
1.Profondeur 2.Loui, A, B, C	Loui	Profondeur Loui	Profondeur	1.Profondeur 2.Loui 3.Largeur

3- Pour le graphe quelconque:

collective			individuelle		
messages	temps		messages	temps	
	Async	Sync		Async	Sync
Profondeur	Largeur	Profond Largeur	Profondeur	Profondeur	Largeur

En plus de la classification, l'étude comparative a permet de conclure que:

1. L'algorithme très simple de Chang et Roberts [CHA 791 qui était à l'origine de la technique d'extinction selective, a une bonne performance compare aux autres algorithmes qui ont été conçus plus tard et dont la mise en oeuvre est bien plus delicate ou même parfois complexe([DOL 821, [HIR 801. Mis à part le cas extreme où l'anneau est decroissant, cet algorithme est le plus interessant de par sa simplicité.

2. Dans le cas de la topologie à maillage complet, l'algorithme(corrigé) de Loui [LOU 861 qui est base sur le sens d'orientation global, donne de bon résultats par rapport aux algorithmes A, B et C de Afek [AFE 85] qui utilisent la notion de 'niveau' représentant le "travail" effectue par les processus participant a l'election.

3. En tenant compte de toutes les configurations de reseau étudiées il apparait que l'algorithme d'exploration en profondeur est le plus performant.

4. Concernant les deux algorithmes conçus pour des topologies quelconques, nous avons observé que, dans certains cas, le calcul d'election base sur une exploration en profondeur est plus rapide en temps d'election que celui base sur l'exploration parallele. Le phénomène à été explique par le fait que dans le calcul parallele, la generation excessive de messages influence de maniere negative le temps d'election.

Les resultats obtenus permettent d'affirmer que si les evaluations de complexités theoriques sont necessaires, elles ne suffisent pas, pour autant, a faire un choix de mise en oeuvre dans un cadre reel, et qu'elles doivent, par consequent, Ptre complétées par d'autres méthodes de validation-evaluation parmi lesquelles la simulation occupe une place importante.

Le choix d'un algorithme d'election depend, principalement, de la nature du problème a résoudre, et de l'importance que l'application donne au trafic ou au temps de reponse. La simplicité de mise en oeuvre peut, aussi, constituer un critère de choix non negligeable.

Le travail réalisé dans cette these est loin d'avoir cerné tous les problemes lies à ce type d'algorithmes. D'autres parametres pourraient être étudiés, tels que l'influence de la longeur des messages sur les performances.

Les algorithmes synchrones qui sont encore peu nombreux, n'ont pas été évalués. Une etude appropriée de leurs propriétés est encore a réaliser.

Il serait, également, interessant et instructif de completer cette etude par l'experimentation de ces algorithmes dans un environnement d'execution réel[BAD 911.

La mise en evidence des techniques de base sur lesquelles sont bases les algorithmes d'election pourrait constituer le point de depart d'une etude plus approfodie de ces algorithmes et d'autres algorithmes de contrôle distribue. Elle permettrait de degager les points communs a ces differents algorithmes, et de mettre au point une methodologie pour leur conception.

B I B L I O G R A P H I E

[ADA 88a] Adam, M.
Exerimentation on parallel machines in helpful to analyse distibuted algorithms / Ph. Ingels, C. Jard, J.M. Jezequel, M. Raynal. - Workshop on "parallel and distributed algorithms", France, (September 1988).

[ADA 88bl Adam, M.
Algorithmes distribues synchrones et systemes repartis asynchrones: concepts, mise en oeuvre et experimentation/P. Ingels, M. Raynal.-IRISA, PI n°411, (Juin 1988).-p 28.

[AFE 851 Afek, Y.
Distributed algorithms for election in unidirectional and complete networks.- University of Califrnia: these de docteur, Inovembre 1985).

[ATT 87] Attiya, H.
Constructing efficient election algorithms from efficient traversal algorithms.- (Juillet 19871.

[ATT 89] Attiya, H.
Efficient election chordal ring networks/ J.V. Leeuwen, N.Santoro, S. Zaks.- Algorithmica, (1989).- pp 437-446.

[ANG 871 Angluin,D.
Local and global properties in networks of processes. -Proc. 12th ACM symposium on theory of computing, (April 19801.-pp 82-93.- In [LAV 871.

[AUD 881 Audureau, E.
Theorie de la programmation et logique temporelle, seconde partie: validation d'algorithmes parallèles/ L.F. Del Ceno, P. Enjalbert.-TSI, (1988).-pp 181-200.

[BAD 911 Badache, N.
Une machine virtuelle Estelle répartie/ S. Bouallag, N. Semaoune.- Proc. 1st Maghrebin Symposium On Programming and System.- Algiers, (October 21-23 1991). -pp 115-131.

[BAR 871 Baruch, A.
optimal distributed algorithms for minimum weight spanning tree, counting leader election and related problems.- ACM Proc. 9th symp. on theory of computing, networks, (May 1987).-pp 230-240.

[BER 811 Berstein, P.A.
Concurrency control in distributed data base systems/ N. Goodman .-Computing Surveys, vol. 13, 2(June 19811. -pp 185-221.- In [MAD 871.

[BOD 861 Bodlaender, H.L.
New upperbounds for decentralized extrema-finding in a rina of processors.- STACS 86, Lecture notes in C.S. (Springer Verlag), (1986).- pp 119-129.

[BOD 88] Bodlaender, H.L.
A better lower bound for distributed leader finding in bidirectional asynchronous rings of processors.- Info. Proc. Let., vol 27, (19881.

[BOU 88] Bouge, L.
On the existence of symmetric algorithms to find leaders in Networks of Communicating Sequential Process.- Acta informatica, vol 25, (19881, (Springer Verlag) .
- pp 179-201.

[BUD 871 Budkowski, S.
An introduction to Estelle: A specification lanquaqe for distributed algorithms / P. Dembinski.- Computer Networks and ISDN systems, 14 (1987) 3-32.

[CHA 791 Chang, E.J.
An improved algorithm for decentralized extrema-finding in circular configuration of processors / R, Roberts.
- Comm. ACM, vol 22, N05, (may 1979).- pp 281-283.

[CHA 881 Chan, M.Y.
Distributed election in complete networks/ Chin, F.Y.L.
- Distributed computer (Springer Verlaql, (1988).
- pp 19-22.

[COR 811 Cornafion (nom collectifl.
Systemes informatiques répartis, concetps et techniques.
- DUNOD Informatique, (Aout 1984).

[COU 861 Courtiat, J.P.
Estelle: un lanqaqe IS0 pour les algorithmes distribues et les protocoles / P. Dembinski, R. Groz, C. Jard.
- Publication interne nº326, IRISA, (Décembre 19861.

[COU 871 Courtiat, J.P.
Estelle:un lanqaqe IS0 pour les alqorithmes distribues et les protocoles / P. Dembinski, R. Groz, C. Jard.
- TSI, Technique et Sciences Informatiques, (19871.

[DIJ 801 Dijkstra, E. W.
Termination detection for diffusing computation / C.S. Scholten.-Inf. Proc. Letters, vol.. 11,1 (August 1980).-pp 1-4.

[DOL 821 Dolev. D.
An 0(nlogn) unidirectionnal distributed algorithm for extrema-finding in a circle/M. Klawe, M. Rodeh.-journal of algorithms. vol 3. (1982).-pp 245-260.- in [RAY 851

[DUD 871 Duda, A.
Estimating global time in distributed system/ G. Harrus, Y.Haddad, G. Bernard.- In Proc. 7th Int. Conf. on Distributed Computing Systems, Berlin, (1987).

[FRA 821 Franklin. R.
On an improved algorithm for decentralized extrema-finding in circular configuration of processors.
- communication of the ACM. vol 25. nº4.(May 1982).
-pp 336-337.

CFRE 87a] Frederickson, G. N.
The impact of asynchronous cmmunication on the problem of electing a leader in a ring / N. A. Lynch.-Proc. of 16th annual ACM on theory of computing, (19891.
- pp 493-503.

[FRE 87bl Frederickson, G. N.
Electing a leader in a synchronous ring / Lynch A.
-journal of the association of computing machinery.
vol 34. nº1 . (January 1987).-pp 98-115.

[GAF 85a] Gafni. E.
Improvements in the bit complexity of two message-optimal election algorithms.- ACM symp. on PODC, (1985).
- PP 175-185.

[GAF 85b] Gafni. E.
Time and message Bonds for election in synchronous and asynchronous complete networks/ Afek. Y.
-Acm symposium on PODC, (1985).-pp 186-195.

[GAR 82] Garcia-molina, H.
Elections in distributed computing system.- IEEE Transactions on Computer, volt-31, N01, (January 19821.
-pp 48-59.

[GOL 87] Goldreich. 0.
Electing a leader in a ring with link failures /
Shrira L.-Acta informatica. vol 24. (1987).-pp 79-91.

[GRO 87] Groz. R.
Peut on verifier des algorithmes distribues par simulation? .-Algorithmique distribuee, INRIA, la colle-sur-loup (Alpes martimes), (6-10 avril 1987).
-pp 143-145.

[GUS 84] Gusella, R.
Anetwork time controller for a distributed Berkeley UNIX system / S. Zatti.- Distr. Proc. Tech. Comm. Newsletter, vol. 6, nQSI.2, IEEE, (June 1984).-pp 7-15.
-In [GUS 861.

[GUS 851 Gusella, R.
The berkeley UNIX 4:3BSD time sychronization protocol: protocol spécifcation / S. Zatti.- University of California, Berkeley: Report nºUCB/CSD 85/250,(June 851.' -In [GUS 861.

[GUS 861 Gusella, R.
An election algorithm for a distributed clock synchronization program / S. Zatti.-IEEE, (1986).

[HEB 851 Hebuterne. G.
Evaluation de performances des algorithmes distribues. -Actes du premier collogue, An gouleme, (16-18 sept 1985). pp 151-166.

[HEL 86a] Helary. M.
Controlling knowledge transfers in distributed algorithms. application to deadlock detection.-rapport de recherche NQ 493, INRIA, (Mars 19861.

[HEL 86bl Helary. M.
Calcul distribue d'un extremum et du routage associe dans un réseau quelconque / A. Maddi, M. Raynal. - RR Nº516, INRIA, (Avril 19861.

[ISO 891 ISO/IEC JTC l/SC 21.
Working draft for a tutorial for Estelle.- Information retrieval transfer and managemenr for OSI USA (ANSI), Boston, (JUN 19-20, 19891.

[JAR 871 Jard, C.
Introduction à la verification des algorithmes distribués.-Algorithmique distribuee, INRIA, la colle-sur-loup, (Alpes maritimes), (6-10 avril 871. -pp 119-138.

[JAR 891 Jard, C.
un compilateur Estelle multi-processeurs pour l'experi-mentation d'algorithmes distribues sur machines paralleles / J.M. Jezequel.-Publication interne nQ453, (Janvier 891.

[JEZ 89a] Jezequel, J.M.
Outils pour l'experimentation d'algorithmes distribues sur machines paralleles. - These de docteur de l'universite de Rennes, (19 Octobre 89).

[JEZ 89b] Jezequel, J.M.
Un compilateur Estelle multi-processeurs pour l'experi-mentation d' algorithmes distribues sur machines paralleles.- Rennes: IRISA. Publication interne NO 453, (Janvier 19891.

[KAI 88] C. Kaiser,
Election sur un anneau par parcours de reconnaissance Anneau unidirectionnel ou bidirectionnel.- Paris: CNAM, dept. mathematique et informatique, (23 decembre 1988).

[KOR 811 Korach, E.
A probabilistic algorithm for decentralized extrema finding in a circular configuration of processors.- Res. Rep. C. S.- University of Waterloo: Dept. of computer science, (1981).- In [LAV 89C].

[KOR 841 Korach, E.
Tight lower and upper bounds for some distributed algorithms for a complete network of processors / S. Moran, S. Zaks.- ACM, (1984). pp 199-207.

[LAM 781 Lamport, L.
Time, clocks and the orderinq of events in distributed Systems .- ACM, (July 19781, vol 21, number 7.

[LAV 871 Lavault, C.
Un alqorithme d' election entierement distribue et décentralisé dans un système distribue quelconque. - RR N0625, INRIA, (19871.

[LAV 89Cl Lavault, C.
Average number of messages for distributed leader finding in rings of processors.- Padesbour FRG: Lecture notes on theoritical aspects of computer science, (Fevrier 16-18, 1989).-pp 269-281.

[LAV 8911 Lavallee, 1.
Yet another distributed election and (minimum-weight) spanning tree algorithms/ C. Lavault.-RR NQ1024, INRIA, (Avril 19891.

[LAV 901 Lavallee, I.
Efficient routing protocols in nameless networks / C. Lavault.- RR nQ1254, (Juin 19901.

[LEL 771 Le Lann, G.
Distributed systems. Towards a formal approch.-IFIP Congress 1977, Holland.-pp 155-160.-In [RAY 851.

[LOU 861 Loui, M. C.
Election in complete network with a sense of direction/ T. A. Matsushita, D.XB. West.- Information proc. let., vol 22, (1986).- pp 185-187.

[MAD 871 Maddi, A.
Contrôle des transferts d'informations dans les systèmes distribués, application a la detection de l'interblocaqe et à la 1' election dans un reseau quelconque de processus.- Université de Rennes 1: these de docteur en informatique, (Mai 1987).

[MAH 881 Mahmoud Bacha. R.
Etude et implementation d' algorithmes de contrôle distribue: exclusion mutuelle et election.-Alger: USTHB. these de magister.-(Septembre 1988).

[MAS 891 Masuzawa, T.
Optimal fault-tolerant distributed algorithms for election in complete networks with a global sence of directives / N. Nishikoma, K. Haqihora, N. Tokuba.-proc 3rd Int. Wok., Nice, France (1989).- Lecture notes in computer Science.- p 11.

[MAT 89] Matias, Y.
Simple and efficient election algorithms for anonymous networks / Y. Afek.-Lecture notes in C.S., Distr. alg., 3rd Int. Work., Nice, France, (Septembre 26-28, 19891. -pp 183-194.

[NAV 881 Navaratnam, S.
Reliable group communication in distributed systems / S. Chanson, G. Neufeld.-Proc. 8th Int. Conf. Distributed Computing Systems. -CS Press, Los Alanitos, California, order nQ865, (June 88).-pp 439-446.

[PAC 841 Pachl, J.
Lower bounds for distributed maximum-finding algorithms/ E. Korach, D. Rotem.- Journal of the association for computing machinery, vol 31, Nº4 „ (Octobre 1984). -pp 905-918.

[PET 821 Peterson. G. I.
An 0(nlogn) unidirectionnal algorithm for the circular extrema problem.-ACM Toplas. vol 4. nº4, (octobre 1982). -pp 758-762.

[PUJ 871 Pujolle, G.
Reseaux et télématique/ D.Secret, D. Dromard, E.Horlait. -Edition: Eyrolles, Tome I, Tome II, (1987).

[RAY 851 Raynal. M.
Alqorithmes distribués et protocoles.-Eyrolles, (1985).

[RAY 88] Raynal. M.
Synchronisation et contôle des systemes et des programmes repartis / J.M. Helary.- Edition: Eyrolles, (1988).-p 194.

[RAZ 841 Razouk, R.R.
Performence Analysis usinq Timed Petri Nets/C.V. phelps. - 4th IFIP workshop on protocol specification, verification and testing. North holland ed: (Juin 1984). -In [HEB 85].

[SAN 861 Santoro, N.
Guessing games and distributed computations in anonymous networks /J. Van Leeuwen, J. Urrutia, S. Zaks. -Preliminary draft, (March 86).

[SHR 85] Shrira, L.
Methodological construction of reliable distributed algorithms / M. Rodeh.-TR361, computer science dept., Technion, Haifa 32000, (Febrary 1985).- in [GOL 871.

[SIN 801 Sinclair. J. B.
Decentralized extrema-finding in circular configurations of processors / Hirshberg D.S.-Comm. ACM. vol 23. nº 11. (November 1980).-pp 627-628.

[SKY 871 Skym. S.
A distributed Election and Spaning tree algorithms based on depth first search traversals.-Marhus university, computer science department, Denmark, (August 1987).-p11.

[SMI 791 Smith, R.G.
The contact net protocol: High level communication and control in a distributed problem solver.- Huntsville, AL.: in Proc. 1st Int. Conf. Distr. Comput. Syst., (Octtober 1979).-pp 185-192.-In [GAR 821.

[SPI 891 Spirakis. P.
Symmetry breaking in asynchronous rings with O(n) messages / B. Tampakas, T.Athamasios.-Lecture notes in computer science .-Distributed algorithms.- 3rd Int. workshop. Nice. France. (Septembre 1989).-pp 235-241.

[VER 871 Verjus, J.P.
construction et preuve d' algorithmes distribues -algorithmique distribuée.-INRIA, la colle-sur-loup (Alpes maritimes), (6-10 Avril 1987).-pp 147-168.

[YAM 891 Yamashita. M.
Electing a leader when processor Identity Number are not distinct / T. Kameda. -Distributed algorithms 3rd International workshop. proc.-Lecture notes in computer science, Nice, France, (septembre 1989).-pp 304-314.

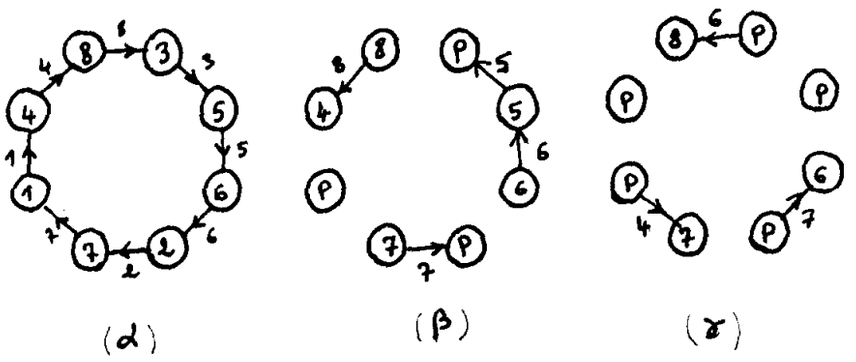
A N N E X E

```
specification S;

default individual queue;

const  nbsites=10;
       nbsites_1=9;

type   numsite=1..nbsites;

channel liaison(sortie,entree);
by sortie:
       elir(numero: integer) ;
       elu(numero: integer) ;
by entree: ;

module type-mod (mon_numero: integer) ;
ip     5: liaison (sortie) ;
       e: liaison(entree);
end;

body type-bod for type_mod;

state participant,non_participant,fin,elu;

var
coordonateur: integer ;

initialize to non_participant
begin
  coordonateur: =0;
end;

trans
from non_participant to participant
(* provided mon_no=10*)
priority 1
begin
  output 5.elir (mon_no) ;
end;

trans
from non_participant to participant
when e.elir (numero)
priority 3
begin
  if numero<mon_numero then output s.elir (mon_numero) ;
                       else outputt s.elir (numero);
end;
```

```
trans
from participant to participant
when e. el ir (numero)
priority 3
begin
   if numero>mon_no then output s. el i r (numero) ;
end;

trans
from participant to fin
when e.elu(numero)
priority 4
begin
   coordonateur: =numero;
   if coordonateur< >mon_no then output s. elu (numero) ;
end;

[illegible]
from participant to elu
when e.elir(numero)
provided (numero=mon_numero)
priority 5
begin
   output a. el u (numero) ;
   coordonateur: =mon_numero;
end;


END; (*endbody*)

MODVAR noeud: array[ 1. .nbsi tes] of type-mod;

initialize
   BEGIN
    ALL i:numsite DO INIT noeud[i] WITH type_bod(i);
    ALL i:1..nbsites_1 DO connect noeudC17.s TO noeud[i+1].e;
    connect noeud[nbsites].s TO noeudCIJ.e;
   end;
END.
```

Exécution de l'algorithme de LOUI (Lou 86] non corrigé :

Proc1	Proc2	Proc3	Proc4	Proc5	Proc6	Proc7	Proc8	
D=?1 E-7	D=?1 E-7	D=?1 E-7	D=??? E=	D=??? E=	D=??14 E=	D=??14 E=	D=??14 E=	
S(1,1,1) R(1,7)	S(1,1,2) R(1,6)	S(1,1,3) R(1,8)	S(1,1,4) R(1,1)	S(1,1,5) WI.31	S(1,1,6) R(1,5)	S(1,1,7) R(1,2)	S(1,1,8) R(1,4)	α
Passif R(7,4)	Passif R(7,7)	Passif R(7,6)	S(7,7,4) R(7,8)	S(7,7,5) R(7,6)	S(7,7,6)	S(7,7,7)	S(7,7,8)	β
S(7,14,4)	S(7,14,7)	S(7,14,6)	Passif	Passif	R(14,7)	R(14,4)	R(14,6)	γ
					Passif	S(-6,-6,7) erreur	S(-6,-6,8) erreur	



Exécution de l'algorithme de LOUI [Lou 86] corrigé :

Proc1	Proc2	Proc3	Proc4	Proc5	Proc6	Proc7	Proc8	
D=?1 E=7	D=?1 E=7	D=?1 E=7	D=?47 E= 2	D=?47 E= 2	D=?? 6 E= 3	D=?? 8 5 E= 5	D=?4863 E=	
S(1.1.1) R(1.7) passif	S(1.1.2) R(1.6) passif	S(1.1.3) R(1.8) passif	S(1.1.4) R(1.1) actif	S(1.1.5) R(1.3) actif	S(1.1.6) R(1.5) actif	S(1.1.7) R(1.2) actif	S(1.1.8) R(1.4) actif	α
R(7.4)	R(7.7)	R(7.5)	S(7.7.4) R(7.8)	S(7.7.5) R(7.6)	S(7.7.6)	S(7.7.7)	S(7.7.8)	β
S(7.6.4)	S(7.6.7)	S(7.6.5)	Passif	Passif	R(6.7)	R(6.4)	R(6.5)	γ
			R(2.7)	R(t.8)	passif	S(2.2.7)	S(2.2.8)	θ
			S(1.3.7)	S(1,3,8)	R(3.8)		R(3.7)	λ_1
					S(2.5.8)	B(5.8) passif	S(5.5.8)	λ_2
						R(5.8) S (3.0.8)	R(0.8) Elu	λ_3