MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

INSTITUT NATIONAL D'INFORMATIQUE
OUED-SMAR

MEMOIRE POUR L'OBTENTION DU DIPLOME D'INGENIEUR D'ETAT EN INFORMATIQUE

Option

SYSTEMES INFORMATIQUES

Thème

EXCLUSION MUTUELLE ET MOBILITE DES HOTES DANS LES SYSTEMES DISTRIBUES

Encadré par:

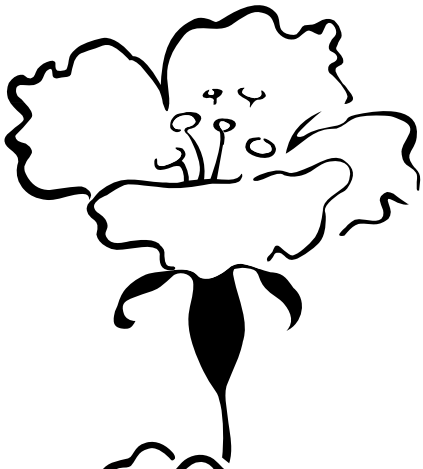
M^r F.Dahamni

Réalisé par:

Djeffal Hamid

Derraa M^ed Bachir

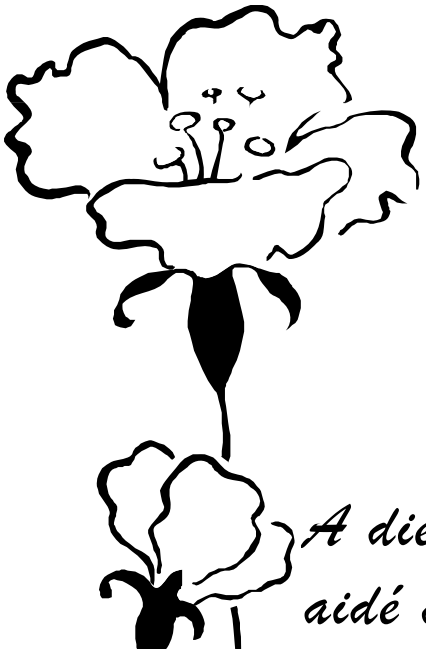
Promotion 1996-1997



Dédicaces

A mes chères parents qui ont tout donné pour que ce jour arrive et sans qui je n'aurais jamais abouti,
à mon frère Karim,
à mes deux soeurs Radia et Nahla,
à la mémoire de mes grands parents,
aux miens,
à tous mes amis,
et à tous ceux avec qui j'ai passé

Bachir



Dédicaces

A dieu le tout puissant qui nous a aidé à faire ce travail,
à mon cher père qui a tout donné pour que ce jour arrive,
à ma chère mère qui a tout sacrifié pour moi,
à la memoire de mon grand père.
à ma grand mère,
à mes frères et soeurs,
à mon oncle,
à tout mes amis,...

je dedie ce memoire.

Abstract

The integration of mobile/portable computers within existing static networks introduces a new set of issues to distributed computation: A mobile host can connect to the network from different locations.

Distributed algorithms, therefore, can not rely on the assumption that a participant maintain a fixed and universally known location in the network. Mobile hosts communicate with the rest of the network via a wireless broadcast medium (limited bandwidth) and often operate in a disconnected or doze mode. All these parameters need to be considered as well, while designing distributed algorithms for such systems.

In this work, we study the impact of hosts mobility and the physical constraints of the mobile hosts on the mutual exclusion problem. After, we suggest two algorithms; each for a specific case of this problem: the channel allocation and the access to a critic section. Finally, we present a mechanism for the simulation of this environment and the test of these algorithms.

Résumé

L'intégration des ordinateurs mobiles/portables dans les réseaux distribués actuels crée de nouveaux problèmes pour l'informatique distribuée: Un hôte mobile peut se connecter au réseau à partir de différentes localités. Par conséquent, les algorithmes distribués ne pouvent plus supposer que la localité d'un site est fixe et universellement connue dans le réseau. Aussi, les mobiles communiquent avec le réseau via une liaison sans fil (de bande de fréquence limitée) et opèrent souvent en mode déconnecté ou en mode veille. Tous ces paramètres doivent être considérés lors de la conception des algorithmes pour de tels systèmes.

Dans ce travail, on étudie l'impact de la mobilité des hôtes et leurs contraintes physiques sur le problème de l'exclusion mutuelle et on propose deux algorithmes traitants chacun d'un cas spécifique de ce problème: l'allocation des canaux et l'accès à la séction critique. Enfin, on propose un mécanisme pour la simulation de cet environnement et le test de ces algorithmes.

SOMMAIRE

Table des figures

Table des tableaux

INTRODUCTION GENERALE

INTRODUCTION GENERALE

Avec le développement, sans cesse croissant, de la technologie des ordinateurs portables et des communications sans fil (cellulaire et satellite notamment), le rêve de pouvoir exploiter les services des réseaux distribués au cours des déplacements, tend à voir le jour.

Actuellement, un utilisateur qui dispose d'un portable, souvent presque aussi puissant qu'une station de travail de bureau, peut travailler lors de ses divers trajets, il est cependant restreint dans ses actions, étant fatalement déconnecté de son réseau.

L'informatique mobile (nomade) vise à résoudre ce problème, en permettant aux portables équipés d'une interface de communication sans fil, de maintenir leur connexions réseau au cours de leurs déplacements.

Avec l'informatique mobile, il sera possible à un utilisateur en connexion sans fil avec le réseau d'accéder à son environnement de travail (données, applications, imprimantes, fax,...), de communiquer, de recevoir des notifications d'événements importants quelle que soit sa position géographique et ses déplacements.

Une projection sur le marché faite par Motorola estime qu'en l'an 2000, il y aura plus de 60 millions d'utilisateurs de P.C.N. (Personal Computer Network) aux U.S.A. seule. Tous ces utilisateurs auront besoins d'utiliser les possibilité de l'informatique mobile.

La réalisation d'un environnement mobile, souvent basée sur des P.C. (Personal computer) portables et des réseaux cellulaires, fait face à des contraintes induites par les caractéristiques techniques de ces deux entités.

Les ordinateurs portables quelque soit leur type et leur puissance sont caractérisés par:

-une petite taille,

-une durée de vie des batteries limitée,

-risque de perte, vol, accident,...

Le médium sans fil, aussi, est caractérisé par:

-une faible largeur de bande passante,

-des déconnexions fréquentes mais prévisibles,

-une diffusion naturelle dans chaque cellule.

Ainsi, le réseau englobant ces entités aura une architecture très dynamique qui n'est pas connue dans les réseau distribués actuels.

Pour que l'informatique mobile puisse atteindre ses objectifs, il est indispensable de concevoir des algorithmes distribués qui prennent en compte la mobilité des sites et les contraintes physiques de ce nouvel environnement.

L'un des problèmes essentiels rencontré par tout concepteur de systèmes d'exploitation est celui de l'***exclusion mutuelle***, problème qui va faire l'objet de notre présent travail, mais dans un environnement mobile.

Il s'agit, donc, d'étudier le problème d'accès en exclusion mutuelle à une ressource donnée dans l'environnement mobile (canaux de communication sans fil, imprimantes, fichiers, variables,...), et de trouver des algorithmes qui, en même temps, garantissent l'exclusion mutuelle et respectent les caractéristiques de l'environnement mobile.

Ce rapport sera organisé comme suit

1.Etude de l'environnement mobile: Dans ce chapitre nous étudierons les caractéristiques physiques de cet environnement, en évaluant son impact sur les différentes entités qui y sont utilisées.

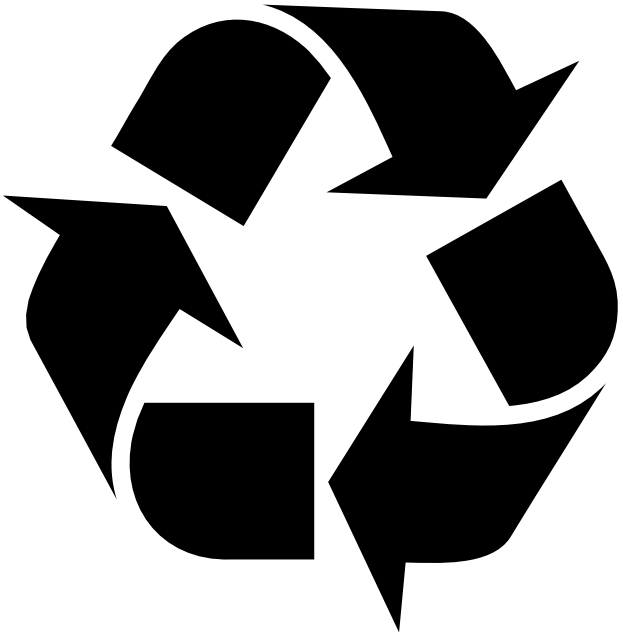
2.Etude des algorithmes distribués d'exclusion mutuelle: Dans ce chapitre, nous présenterons les différents algorithmes appliqués dans les systèmes distribués statiques et nous étudierons les différentes techniques qu'ils utilisent. Enfin, on discutera l'éventuelle adaptation de ces algorithmes à l'informatique mobile.

3.Choix des algorithmes pour l'environnement mobile: Ce chapitre sera consacré au choix des algorithmes les plus adaptés en essayant d'y apporter les améliorations nécessaires.

4.Mise en oeuvre: Dans ce chapitre, on essayera de trouver un moyen pour la simulation de l'environnement mobile et de test des différents algorithmes conçus pour cet environnement.

CHAPITRE I

ETUDE DE L'ENVIRONNEMENT MOBILE



CHAPITRE I: ETUDE DE L'ENVIRONNEMENT MOBILE

I.INTRODUCTION

Le développement de la technologie des communications sans fil via réseaux cellulaires et satellites promet d'offrir, aux utilisateurs des réseaux distribués informatiques, la possibilité d'accès aux réseaux de n'importe quel endroit à n'importe quel moment.

Dans un futur proche, des dizaines de millions d'utilisateurs, possédant des ordinateurs portables équipés d'interfaces de communication sans fil, pourront exploiter les services réseaux sans aucune restriction sur leur localité [IMI, 94]. L'introduction de cette approche va créer de nouvelles possibilités et diverses applications dans le domaine des réseaux informatiques.

La mobilité des hôtes ainsi que leurs caractéristiques matérielles influeront fortement sur l'architecture, le système et les applications de ces réseaux.

Dans cette partie, nous allons présenter ce nouvel environnement avec ses réseaux, son matériel et ses systèmes d'exploitation.

II.PRESENTATION DE L'ENVIRONNEMENT MOBILE

II.1.Définition de l'environnement mobile

L'environnement mobile est l'ensemble matériel et logiciel qui permet à un utilisateur de réseau de communiquer sans aucune restriction sur son mouvement. Il est indispensable, pour réaliser un tel environnement, de disposer des éléments suivants:

-Des machines transportables (petites et légères) et qui peuvent envoyer et recevoir des messages.

-Une liaison de communication permettant la liberté de mouvement aux utilisateurs.

-Les mécanismes systèmes nécessaires à la gestion d'un tel environnement.

-Un réseau pouvant supporter la mobilité.

II.2.Définition de la mobilité

La notion de mobilité, dans le domaine de la télécommunication, représente l'aptitude à communiquer tout en se déplaçant, c-à-d avoir une totale liberté de mouvement tout en gardant la liaison avec le réseau. Cette mobilité n'était pas possible dans les anciens réseaux de communication, où pour communiquer il fallait avoir deux machines reliées par une liaison filaire. La réalisation de machines mobiles nécessite l'élimination de toute liaison filaire, que ce soit pour la communication ou pour l'alimentation qui pourrait restreindre leur mobilité.

L'élimination de la connexion d'alimentation nécessite l'utilisation de sources d'énergie transportables (les batteries). Cependant, quelle que soit la quantité d'énergie offerte par ces dernières, leur durée de vie est limitée et par conséquent, la nécessité du traitement prudent de cette ressource s'impose. Sa conservation peut se faire sur les deux plans matériel et logiciel.

En plus de la connexion d'alimentation, la mobilité exige l'élimination de la liaison filaire de communication, ce qui conduit à l'utilisation de moyens de communication sans fil.

II.3.Connexions sans fil

L'utilisation de ce type de connexions permet de libérer l'utilisateur de la liaison de communication filaire. Et par conséquent, lui permettre d'accéder au réseau sans prendre en compte ni sa localité ni son mouvement durant la communication.

Ces connexions ne sont pas toujours fiables. D'une part, elles offrent un débit plus bas que celui offert par les connexions filaires, et d'une autre part, elles sont sensibles aux différentes perturbations.

Le tableau suivant présente les différents types de connexions sans fil ainsi que leurs caractéristiques, avantages et inconvénients:

		Longueur d'onde	Fréquence utilisée (MHz)	Modulation	Avantages	Inconvenients
Ondes lumineuses	Infrarouge	1-100µm	10^{12}-10^{14}	Généralement temporelle	-Fiable	-Distance limitée -Obstacles bloquants -Consommation d'énergie élevée
	Laser	1000A-1µm	10^{14}-10^{16}	Généralement temporelle	-Immunité aux perturbations -Haut debit	-Visibilité directe -Consommation d'énergie élévée
Ondes radio-fréquences	Ondes radio	>10cm	$<10^{9}$	fréquence ou phase	-Traverse les obstacles	-Sensible aux perturbations -Legislations contrainiantes
	Micro-ondes	100µm-10cm	10^{9}-10^{12}	Fréquence	-Immunité aux perturbations -Haut debit	-Faible distance

Tableau 01:Types des connexions sans fil

L'utilisation des ondes lumineuses, dans la télécommunication, nécessite que l'émetteur et le récepteur soient en vue l'un de l'autre, ou qu'ils exploitent les propriétés de réflexion de ces médiums. Ajouté à cela, la consommation d'énergie des transmissions par ondes lumineuses est très élevée, ce qui épuise la source d'énergie qui alimente le mobile.

Contrairement aux ondes lumineuses, les ondes radiofréquences traversent les différents obstacles (sauf charpentes métalliques) et peuvent atteindre de grandes distances (selon la puissance et la fréquence utilisée) ce qui permet une très grande liberté de mouvement de ses utilisateurs.

Le seul problème présenté par ce type de connexions, est qu'elles sont limitées par la législation des télécommunications, c-à-d la bande de fréquence réservée pour une application mobile est relativement réduite, mais une bonne gestion de cette bande peut remédier à cet inconvénient.

II.4.Machines mobiles

Pour permettre la mobilité d'une machine, il faut, comme il a été déjà précisé, la libérer de ses liaisons d'alimentation et de communication et miniaturiser ses composantes pour permettre sa portabilité.

L'utilisation des batteries a permis d'éliminer la liaison de l'alimentation. Selon leur durée de vie - le temps d'autonomie (stand-allone) permis au mobile utilisant cette batterie -, leur poids, et leur taille, les batteries diffèrent.

Le tableau suivant présente les grands types de batteries se trouvants actuellement sur le marché:

Type	Capacité (Volts)	Avantages	Inconvénients	Utilisation
Les batteries *Nickel-Cadium* (NiCd)	1.2	Moins coûteuses	Problème de mémoire de charge[1]	Diverses
Les batteries *Nickel-Métal-Hybride* (NiMH)	1.2	Capacité élevée à poids égal	Coûteuses et fragiles	Diverses
Les batteries *Plomb-Gelifié* (Pb)	2	Faible autodécharge	charge relativement lente	Systèmes de sécurité
Les batteries *Lithium-Ion* (Li)	4	Puissantes	Très chères et polluantes	Domaine spatial et militaire

Tableau 02: Différents types de batteries

Pour augmenter le plus possible la durée de vie de ces batteries, plusieurs mécanismes sont envisageables:

-Réalisation d'éléments dont la consommation d'énergie est minimale. Dans ce sens, Intel a proposé le processeur SL, CYRIX et AMD les processeurs SLC et SXLV. Ces processeurs fonctionnent sur 3.3 Volts au lieu de 5 Volts et disposent de fonctions de gestion de la consommation d'énergie pour travailler à des fréquences réduites si l'application le permet.

- Permettre aux applications des opérations de conservation d'énergie telles que l'arrêt du disque, l'extinction de l'écran, et des opérations de réduction des besoins de calcul et de communication et aussi la fréquence d'exécution des actions périodiques pour les surcoûts dus à leur lancement.

Il va être utile aussi d'introduire l'opération de mise en veille qui permet à un portable en attente de suspendre toutes les exécutions et de réduire la vitesse de l'horloge. Il pourra revenir à son ancien état dès que la cause de l'attente se termine. Par exemple, un mobile demande une exécution

[1]*Le phénomène de mémoire de charge se présente comme suit:si une batterie NiCd n'est pas totalement chargée ou qu'elle soit chargée avant d'être complètement déchargée, elle mémorise ces deux niveaux comme étant ses capacités minimale et maximale et ainsi perd de son efficacité.*

à distance sur un serveur, en attendant que les résultats lui soient transmis, il pourrait se mettre en veille.

La réalisation de ces mécanismes nécessite la revue des protocoles de communication (contrôle des erreurs, compression, cryptage,...etc.), et des algorithmes constituants les systèmes d'exploitation gérant ces mobiles (exclusion mutuelle, élection, interblocage,...etc.). Dans cette nouvelle conception, il faudra prendre en compte la consommation d'énergie des différents éléments des portables.

Pour des raisons de portabilité, les éléments constituants un portable sont miniaturisés: le poids des machines actuelles est en moyenne de 2 Kg et leur taille ne dépasse pas, généralement, le format A4.

Cette miniaturisation était au détriment des capacités de calcul et de stockage des machines portables. Pour remédier à ces inconvénients, des mécanismes systèmes sont envisageables et qui seront étudiés par la suite. Il faut aussi adapter les applications conçues pour des machines fixes (de bureaux) aux machines mobiles pour éviter leur dégradation. On pourrait également fournir des services de calculs à distance (l'accès aux serveurs de calcul).

II.5.Réseaux mobiles

Un réseau mobile est un réseau qui permet à ses utilisateurs de se déplacer en cours d'accès. Les réseaux mobiles diffèrent selon le degré de liberté de mouvement et les services réseaux qu'ils offrent aux utilisateurs.

II.5.1.Téléphonie cellulaire

La surface couverte par un tel réseau est divisée en sous-surfaces, disposant chacune d'une station fixe MSS (*Mobile Service Station*) gérant les communications sans fil à l'intérieur de cette zone. Ces sous-surfaces de forme circonférencielle (360°), et de superficie limitée par la portée d'émission et de réception des MSSs et des téléphones, sont appelées cellules. En considérant les limites entre ces zones, on aboutis a des cellules de forme hexagonale (figure 01). Les stations, gérants ces cellules, représentent, pour les téléphones mobiles, les points d'accès au réseau. La gestion des déplacements des téléphones entre les cellules est sous la responsabilité du réseau, elle est assurée par des commutateurs généraux et régionaux.

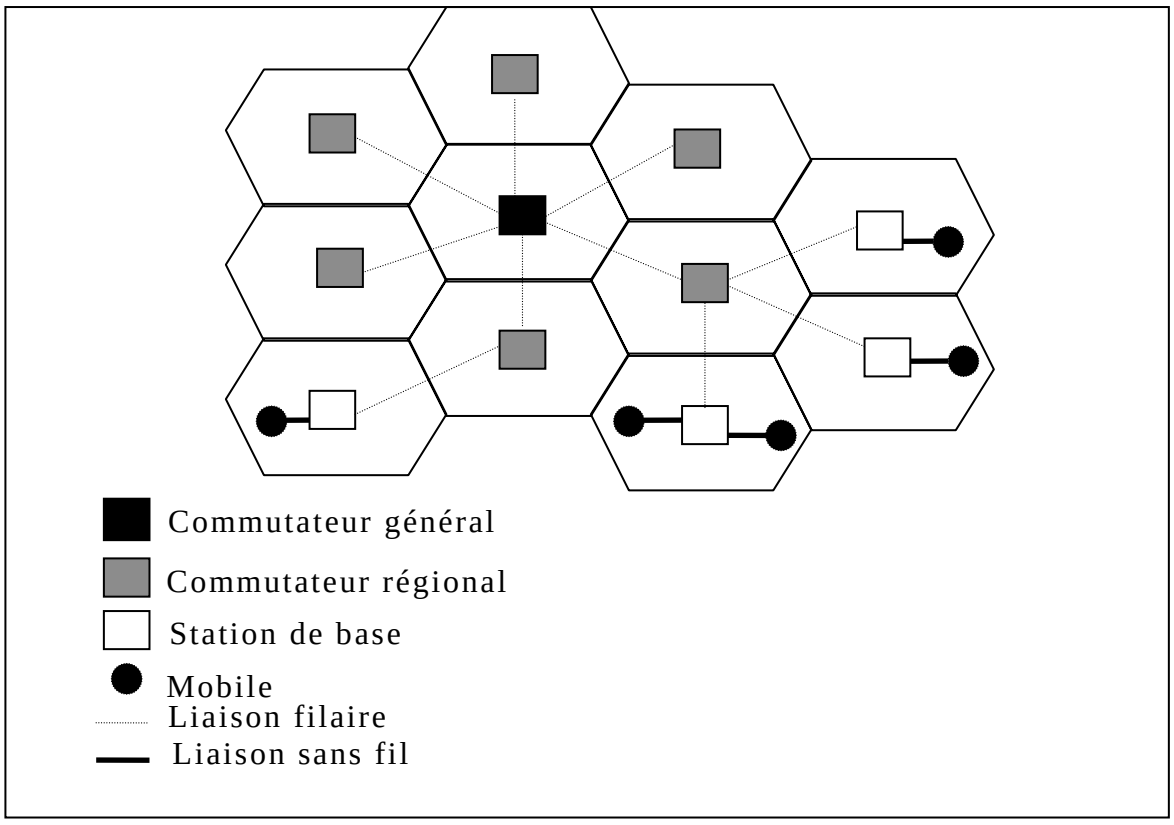


Figure 01: Architecture de la radio téléphonie cellulaire [Bag, 95]

II.5.2.Les LAN sans fil

C'est un *LAN* (Local Area network) traditionnel (*ETHERNET* par exemple) équipé d'une interface de communication sans fil. Le *LAN* sans fil est en plus connecté à un réseau fixe (plus chargé) comme le *LAN*, le *WAN* (Wide Area Network), *INTERNET*, ... etc.

La composante principale dans un *LAN* sans fil est la carte d'interface sans fil qui possède une antenne. Cette carte peut être connectée à l'unité mobile comme au réseau fixe. Les *LAN* sans fil ont une portée limitée et sont désignés pour être utilisés dans un environnement local seulement (à l'intérieur d'un bâtiment) et par conséquent ne peuvent fournir un support réseau pour les déplacements intercellulaires.

Parmi les *LAN* sans fil disponibles actuellement, il y a *WAVELAN* de *NCR*, *ALTAIR* de *Motorola*, *RANGELAN* de *PROXIM*, *ARLAN* de *TELESYSTEM*.

II.5.3.Les réseaux sans fil à large couverture

Ce sont des réseaux spéciaux de radio-portables fournis par des fournisseurs de services privés tel *RAM* et *ARDIS*.

L'infrastructure fournit une couverture à largeur nationale pour des services de données à faible largeur de bande. Par exemple le réseau *RAM* offre un service de courrier sans fil à un niveau national, et *ARDIS* fournit un accès sans fil à partir d'une unité portable à une application s'exécutant sur un hôte fixe. Il n'y a pas de support pour le changement des connexions dans le réseau et il n'est pas clair comment le réseau va gérer le changement du nombre massif des mobiles dans sa couverture.

II.5.4.Le réseau *Pagers*

Dans ce réseau la couverture n'est pas un problème mais le service est usuellement réception seulement et a une faible largeur de bande. Comme exemple il y a *SKYTEL* de *Motorola*.

En plus, de nouveaux services satellites sont proposés : *IRIDIUM* de *Motorola* avec 66 satellites qui tournent autour de la terre sur des orbites basses *LEO* (*Low Earth Orbit*) est l'un d'eux.

Les premières applications prédominantes pour les systèmes satellites sont la voix, *paging*. D'autres services ont été ajoutés tel que la messagerie et les transmissions fax.

II.5.5.Modèle d'un réseau mobile

Après ce survol des architectures de communication sans fil existantes, on voit bien qu'une architecture finale pouvant supporter l'informatique mobile (transmission de données, communication informatique,...) est loin d'être réalisée.

La plupart des architectures proposées, actuellement, sont de type cellulaire. Dans cette architecture, le réseau est constitué d'un ensemble de stations fixes et un ensemble de stations mobiles.

La partie fixe est le réseau reliant les stations fixes, équipées d'interfaces de communication sans fil qui servent comme point d'accès aux stations mobiles. Ces stations fixes, souvent appelées MSS (*Mobile Service Station*), sont réparties sur l'ensemble des cellules pour gérer l'espace géographique couvert par celles-ci.

La partie mobile est un ensemble d'hôtes mobiles (portables) équipés eux aussi d'interface de communication sans fil. Un hôte ne peut accéder au réseau qu'à travers la MSS gérant la cellule où il se trouve (figure 02).

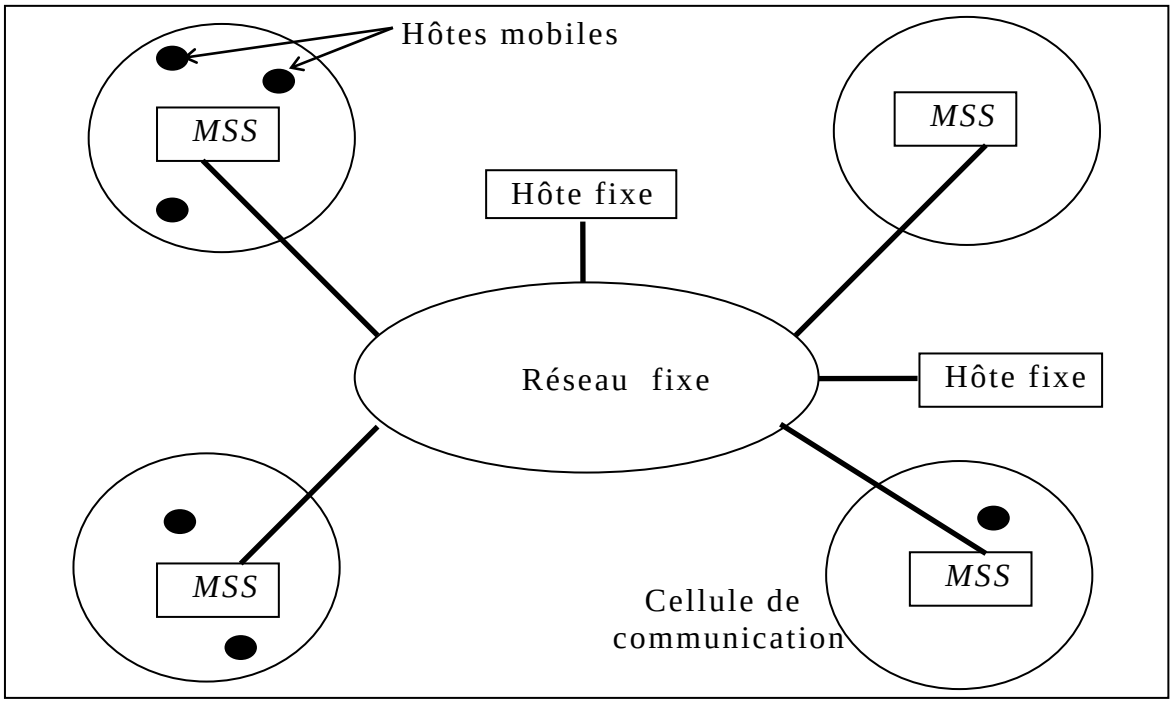


Figure 02 : Modèle d'un système supportant la mobilité [IMI, 94]

Le schéma qui suit (figure 03) nous montre les détailles de la liaison entre une MSS et un mobile:

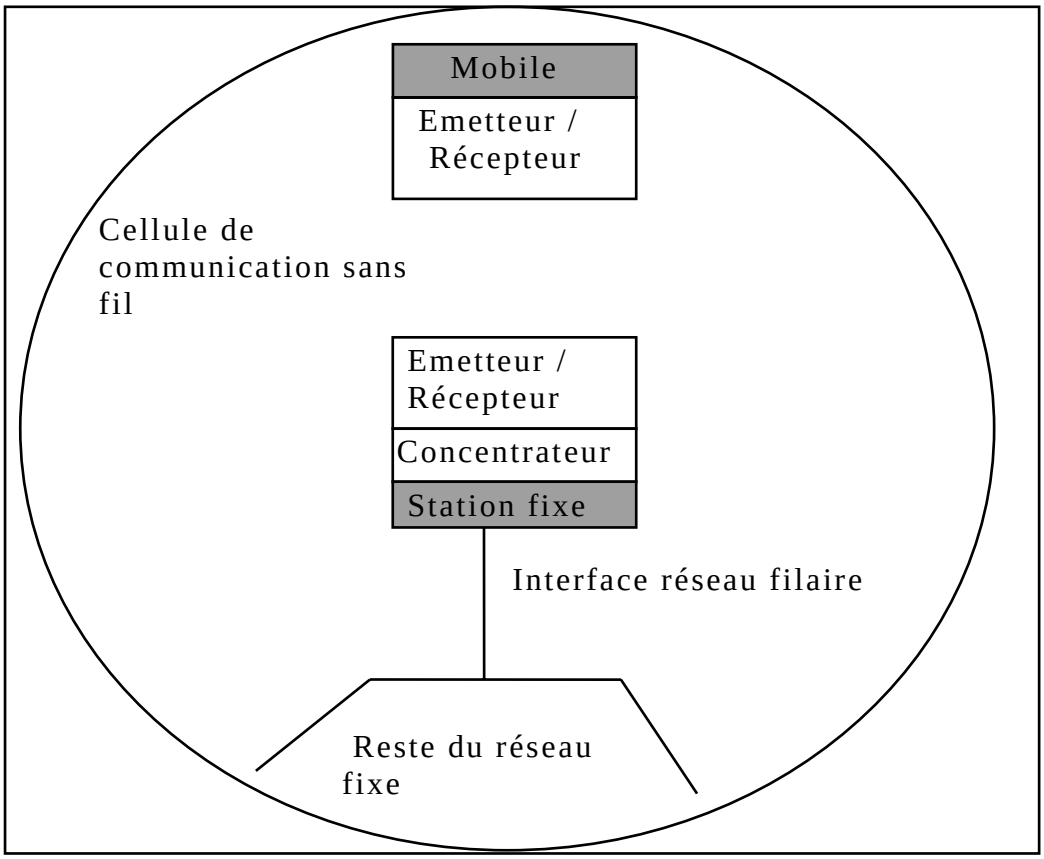


Figure 03: Architecture d'un réseau sans fil [Bag, 95]

Divers problèmes sont apparus lors de l'implémentation de ce schéma (figure 02), entre autres l'allocation des canaux de communication aux mobiles, le basculement des mobiles entre les cellules de communications (*Hand-off*), la localisation des mobiles,... etc.

II.5.5.1.L'allocation des canaux

Dans un réseau cellulaire, l'utilisation du même canal radio pour des communications dans deux cellules adjacentes poses des problèmes d'interférences. Pour éviter un tel problème, on a besoin d'une stratégie pour distribuer l'ensemble des canaux de communication disponible entre les différentes cellules, de façon à ce que cette distribution permette de maximiser le nombre des sessions de communication que peut supporter le réseau entier sans pour autant provoquer des interférences. Cela peut être atteint en exploitant au maximum le principe de réutilisation de la même

fréquence (*Frequency reuse*: utilisation du même canal dans les cellules disjointes).

L'utilisation de la même fréquence (canal) pour supporter des communications dans deux cellules adjacentes pose des problèmes d'interférence aussi bien pour les mobiles que pour les stations de base gérants ces cellules. Pour éviter ce problème et maximiser le nombre total de communications dans le réseau entier, on peut utiliser les mêmes fréquences dans des cellules disjointes (non adjacentes). La distance entre ces cellules nous permet d'éviter toute interférence. Selon la méthode d'affectation des canaux disponibles aux différents cellules, on distingue deux approches, statique et dynamique:

II.5.5.1.1.Méthode statique (Fixed Channel Allocation strategy)

Le spectre de communication est divisé en canaux qui sont affectés aux cellules d'une manière fixe et qui évite toute interférence. Un mobile qui se trouve dans une cellule utilise un canal réservé à cette cellule pour communiquer avec sa station de base.

L'inconvénient de cette méthode se manifeste lors de la concentration d'un nombre élevé de mobiles dans la même cellule. Dans ce cas, le nombre de canaux affectés à cette cellule ne suffit plus pour satisfaire les demandes de session de communication (saturation) alors que, peut être, dans d'autres cellules il y à un grand nombre de canaux libres. Ceci augmentera le temps de latence des mobiles demandants une session de communication et par conséquent provoquera une dégradation de la qualité de service.

L'avantage de cette méthode réside dans sa simplicité.

II.5.5.1.2.Méthode dynamique (Dynamic Channel Allocation strategy)

Dans cette stratégie l'ensemble des canaux réservés à une cellule varie avec le temps. Si une cellule est surchargée et l'ensemble de ses canaux ne suffit plus, elle peut en emprunter à ses voisines de manière à ce que cet emprunt ne conduit pas à des interférences entre les cellules adjacentes. Ces techniques nécessitent l'utilisation de dispositif matériel ou logiciel pour permettre le déroulement des protocoles de changement des canaux.

Il est à noter qu'un certain nombre de canaux est réservé aux messages de contrôle entre les mobiles et les stations de base. Ces messages de contrôle

peuvent être de type demande: d'accueil, d'un canal de communication, de déconnections, de reconnection, … etc.

II.5.5.2.Changement de cellules (Hand-Off)

Pour permettre un service continu aux utilisateurs, le réseau cellulaire doit disposer de mécanismes pour gérer le changement des cellules (Hand-Off). Pour pouvoir contrôler un tel basculement, il faut disposer d'un mécanisme de contrôle permanent de la liaison entre un mobile et la MSS de la cellule où il se trouve, et selon la qualité du signal échangé avec cette station, le mobile décide s'il faut effectuer un basculements vers une autre cellule ou non. Si de telles conditions sont vérifiées, le mobile envoie une demande d'accueil à une autre station (de la cellule où il vient d'entrer). La MSS qui reçoit cette demande communique avec l'ancienne pour réaliser le basculement (transfert des informations concernant le mobile tel que son identité, son serveur habituel, ses fichiers,…etc.).

La superficie d'une cellule est choisie selon la portée du signal émis par le mobile, et de façon à ce que le temps de traversée de cette cellule par un mobile ne soit pas plus court que le temps de gestion du Hand-off sinon on risque de ne jamais pouvoir atteindre ce mobile.

La gestion du Hand-off doit être la plus transparente possible pour les utilisateurs, c-a-d minimiser le temps d'exécution des protocoles gérant le transfert des paramètres et des données des mobiles à la nouvelle cellule pendant le Hand-off pour continuer l'accès au réseau.

Dans ce cadre, il va être utile de dupliquer les données des mobiles dans les stations de base des cellules voisines à celle où le mobile évolue. En somme, cela réduit le Hand-Off à un simple changement d'adresse, mais risque de créer des problèmes de congestion.

Une autre méthode repose sur le principe des groupes de communications [Bir, 94], où on réduit la taille de la cellule de façon à ce que l'émission du mobile peut atteindre plusieurs stations de bases. Ces stations font le même travail pour le mobile mais une seule (appelée primaire) lui envoie des messages. Si le signal reçu par la station primaire devient de qualité inférieure à celle du signal reçu par une autre station , alors cette station jouera le rôle de la station primaire.

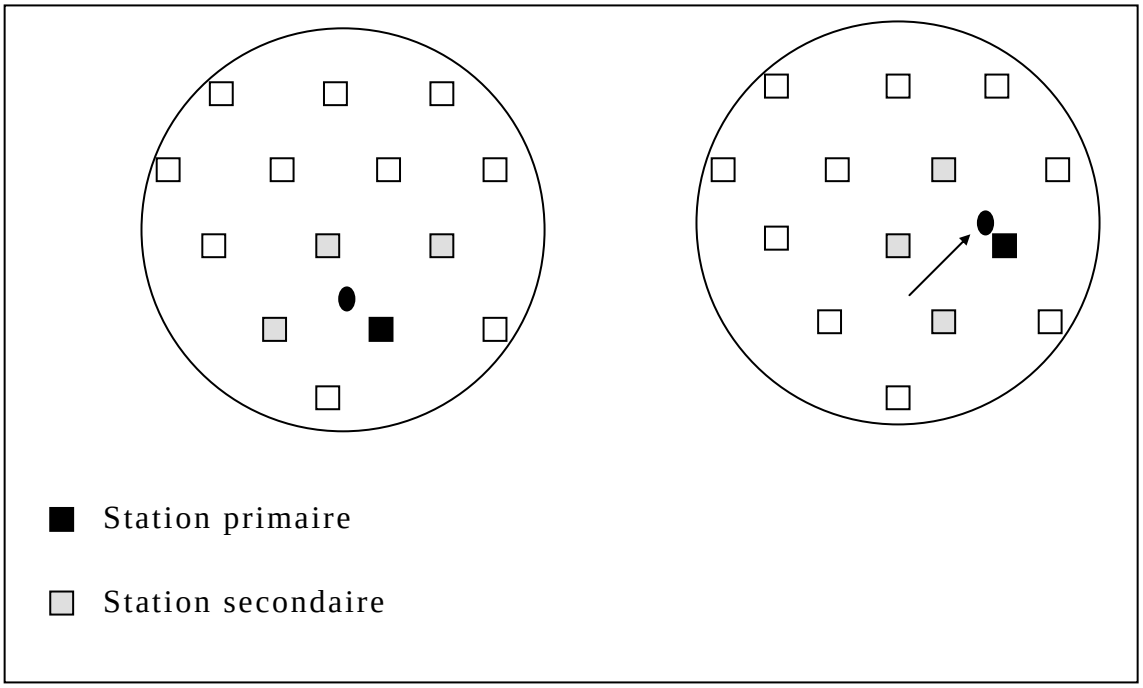


Figure 04: Gestion des Hand-off par les groupes des communications [Bir, 94]

Cette approche, en premier lieu, a été conçue pour réaliser une tolérance aux pannes, ensuite on a vu qu'elle s'adapte bien à la gestion des Hand-offs et permet l'optimisation du temps de leur traitement en minimisant le transfert des données et en le réduisant à un simple changement de nom (station primaire). Ainsi elle permet de tolérer les pannes des stations de base et des liaisons sans fil.

L'inconvénient de cette méthode est le traitement inutile fait par des stations pour des mobiles qui ne vont peut être pas les visiter.

II.6.Comportement d'un mobile dans un réseau

Dans un environnement statique, les déconnections d'une machine du réseau sont, généralement, dues aux pannes et sont imprévisibles et par conséquent traitées comme des cas particuliers. Par contre, dans un environnement mobile, ces déconnections sont très fréquentes à cause de la sensibilité de la connexion sans fil aux perturbations et à la faiblesse de la source d'énergie des mobile.

Il sera comme même irréaliste de considérer les déconnexions fréquentes et généralement prévisibles des mobiles comme des pannes et des

cas particuliers. Pour ces raisons, trois modes d'opération pour un mobile, dans un réseau, sont proposés:

II.6.1.Mode connecté

C'est le cas de la liaison normale entre le mobile et la station de base.

II.6.2.Mode déconnecté

C'est le cas où il n'y a plus de liaison physique entre le mobile et le réseau, soit volontairement ou non. Dans le premier cas, l'utilisateur demande de couper la connexion avec le réseau et un protocole de déconnection est exécuté entre le mobile et la station. L'utilisateur continue, ensuite, ses exécutions sur sa machine. Pour le réseau, cet utilisateur est considéré déconnecté et ne peut être atteint. Les messages qui lui sont destinés sont stockés en attendant sa reconnection. Dans le deuxième cas, soit c'est une panne du mobile ou de la station de base, soit c'est une coupure provoqué par de fortes interférences ou par la surcharge de la station de base. Dans ce cas les précautions nécessaires doivent être prise pour permettre la reprise après la reconnexion du mobile.

II.6.3.Mode veille

Comme sur une machine fixe, l'utilisateur d'un mobile peut demander des services aux serveurs informatiques, en attendant, le mobile peut se mettre en veille, c-a-d suspendre ses applications et éteindre ses périphériques pour conserver l'énergie, et ne fait qu'attendre un signal du réseau (pour récupérer ses résultats ou pour les besoins du réseau lui même), à ce moment là, il restaure son environnement normal et se retrouve en mode connecté.

III.IMPACT DE LA MOBILITE

La mobilité des sites, dans les réseaux distribués, n'influe pas seulement sur la taille des machines, les moyens de communication ou d'alimentation mais influe aussi sur l'architecture, les protocoles, et les applications de ces réseaux.

Cette influence peut être vue sur trois niveaux:

III.1.Architecture

La mobilité des hôtes implique un changement de leur points d'accès au réseau, c-à-d que l'architecture du réseau entier change avec leurs mouvements.

Dans les réseaux distribués statiques, des algorithmes reposent sur des structures logiques (par exemple l'anneau logique) indépendamment de l'architecture physique du réseau. A priori, cette méthode paraît efficace pour palier aux problèmes de la mobilité, mais en réalité, même cette structure logique change en fonction des mouvements, déconnections, et reconnections des mobiles (figure 05).

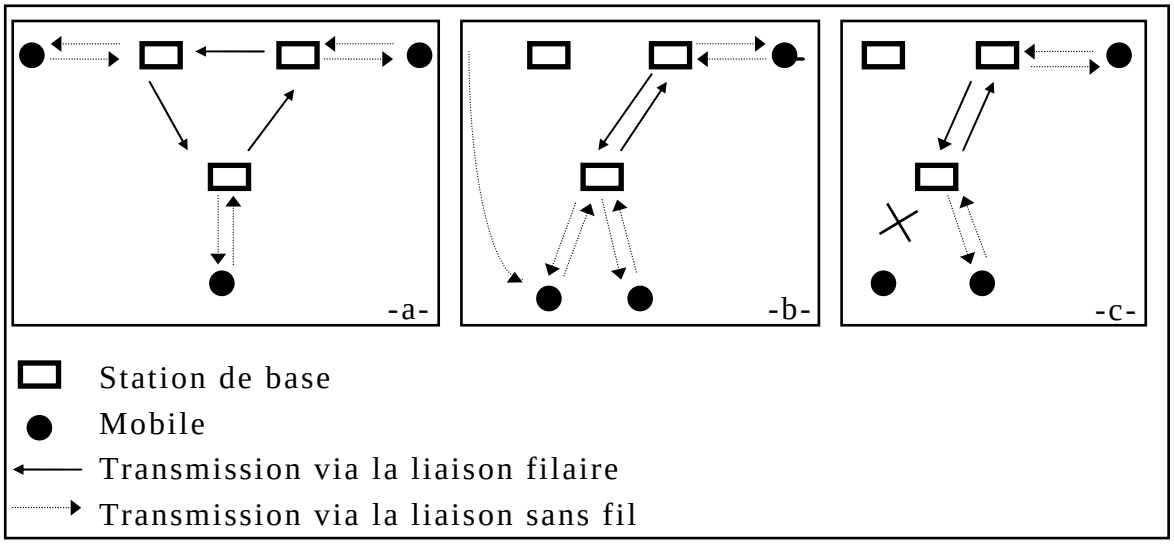


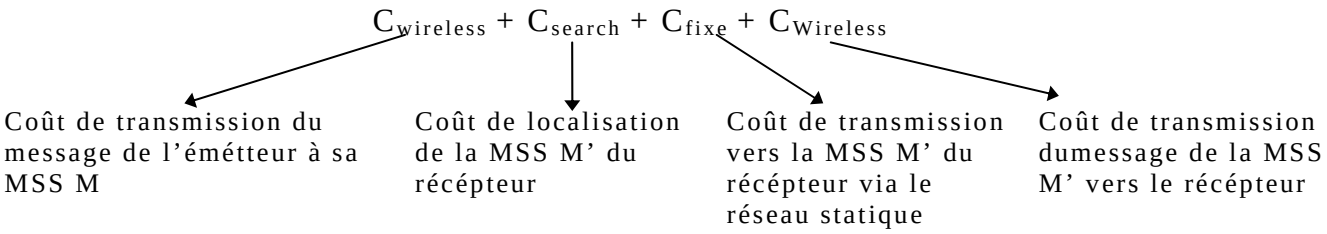
Figure 05: Effets de la mobilité sur la structure d'anneau logique [Bad, 93]

III.1.1.Localisation

Dans les réseau mobiles, la localisation est la base de toute communication avec les entités mobiles, ce qui rend fréquente l'exécution des protocoles qui la réalisent. Pour réduire au minimum cette tâche, une solution optimale s'avère très nécessaire.

Pour qu'un mobile puisse envoyer un message à un autre mobile, il doit l'envoyer à la MSS (M) de la cellule où il se trouve, via la liaison sans fil. Cette MSS lance une recherche, dans le réseau, de la MSS (M') gérant la cellule où le mobile récepteur évolue actuellement. Une fois M' trouvée le message lui est transmis via le réseau statique, pour qu'elle le transmette au

mobile récepteur via la liaison sans fil. Le coût de cette transmission est donc:



Le coût de transmission d'un message ne dépend plus seulement de la distance entre l'émetteur et le récepteur, mais dépend aussi de la stratégie de localisation du récepteur.

Pour pouvoir, donc, acheminer un message à un site mobile, il est indispensable de connaître sa position, c-à-d le localiser. Une première solution consiste à envoyer le message au serveur habituel du mobile qui s'occupe de le lui transmettre. Une autre solution consiste à localiser le mobile destination directement à partir de la MSS gérant le mobile émetteur qui va communiquer avec lui. Mais le problème réside toujours dans la méthode par laquelle le mobile est localisé.

En réalité, il existe plusieurs méthodes de localisation des mobiles , présentant chacune ses avantages et ses inconvénients.

III.1.1.1.La méthode Search

Elle consiste à chercher le mobile dans le réseau entier, c-à-d envoyer une demande de localisation à toutes les MSS, et celle gérant la cellule où le mobile se trouve répond. le coût engendré par cette méthode dépend seulement du nombre des stations de base dans le réseau mobile.

III.1.1.2.La méthode Inform

Elle fait passer le message au mobile à travers son serveur habituel, qui est informé de façon permanente de ses mouvements. Le coût dans cette méthode dépend du nombre des mouvements intercellulaires faits par le mobile.

III.1.1.3.Méthode combinée

Les deux méthodes *Search* et *Inform* peuvent être combiné de façon à ce que le mobile informe son serveur de sa position à chaque intervalle de

temps, ou après un certain nombre de mouvements, et est ensuite recherché dans le voisinage de sa dernière position connue. Evidement, le mobile n'est recherché que s'il n'est pas trouvé dans sa dernière position connue

III.1.1.4.Autre méthode

Dans le cas de concentration d'un grand nombre de mobiles dans un petit area, les serveurs de localité dans cet area peuvent être surchargés voir engorgés par les messages de localisation. Pour palier à ce problème, une méthode dynamique de localisation est proposée, dans laquelle, l'information sur la localité d'un mobile est dupliquée dans un ensemble de serveurs (MSS), qui varient en fonctions de la localité du mobile, pour distribuer la charge de localisation sur tous le réseau [Sin,95].

Evidement, dans toutes ces méthodes le mobile n'est recherché que s'il n'était pas trouvé dans sa dernière position connue.

Le choix entre ces méthodes nécessite une étude du comportement général des mobiles, par exemples, la méthode *search* est moins coûteuse que la méthode *Inform* pour les mobiles qui se déplacent beaucoup, et vice versa. La méthode dynamique est efficace pour les cas de concentration des mobiles dans des zones spécifiques.

III.1.2.Nommage

L'adresse d'un site fixe est, généralement, confondue avec son nom et associée à ses fichiers dans son environnement habituel (ensemble des serveur avec lesquels il communique habituellement), par contre, un site mobile est en mouvement continu (dans son environnement ou ailleurs) et son adresse change suivant ce mouvement.

Il faut, donc, premièrement, séparer le nom qui est fixe d'un site mobile de son adresse qui est dynamique, ce qui conduit à donner un nom à chaque arrivé dans un sous-réseau (où son nom n'est pas connu) et savoir comment utiliser ce nom pour communiquer ou pour accéder aux fichiers associés à son nom ou son adresse habituelle (*home adress*).

III.1.3.Routage

Dans les réseaux distribués statiques, où les sites sont fixes et leurs adresses sont connues d'avance, le routage se fait de diverses manières:

-Router par le plus court chemin.

-Router par diffusion sur plusieurs chemins.

-Centraliser le routage (serveur de routage).

Certains réseaux choisissent un chemin une fois pour toutes lors de l'établissement de la communication (statique), et d'autres le font à chaque envoie d'un paquet (dynamique).

La méthode dynamique paraît être plus adaptée à l'environnement mobile, où l'adresse du destinataire peut changer au cours de la transmission.

La méthode centralisée, utilisant un serveur de routage, n'est pas adaptée à un environnement mobile, car les messages peuvent refaire les mêmes chemins ou faire des chemins inutiles. En général, les autres méthodes sont utiles pour transmettre les paquets d'une MSS à une autre, en tenant compte le fait que le transfert peut être interrompu pour être rediriger vers une autre position du mobile.Entre le mobile et la MSS, on n'a pas besoin de router les messages (paquet) car la MSS représente le seul point d'accès au réseau pour le mobile.

III.2.Communication

Les mobiles communiquent avec le réseau via une liaison sans fil (canal radio) qui est d'une largeur de bande inférieur à celle du médium filaire. Cet inconvénient peut être surmonté en utilisant des techniques de compression de données, en prenant en considération les contraintes d'énergie et des capacités de calcul et de stockage sur les entités mobiles.

III.2.1.Diffusion du médium sans fil

Le médium sans fil supporte la communication par diffusion (*Broadcast communication*) contrairement au médium filaire qui fonctionne usuellement en mode point-à-point. Ces différences entre les deux médiums impliquent la différence dans le coût d'utilisation de chacun.

La communication par diffusion dans le médium sans fil permet d'envoyer un message à plusieurs utilisateurs dans la même cellule en une seule émission, c-à-d que le coût d'une telle émission ne dépend pas du nombre de récepteurs. Cet avantage pose comme même des problèmes à régler au niveau des protocoles qui gèrent une telle communication.

Prenons l'exemple suivant: Une MSS M1, derrière un mobile MH, diffuse un message m, par la méthode de diffusion dans toutes les cellules,

aux mobiles MH1,MH2,MH3 qui se trouvent respectivement dans les cellules C1 (gérée par M1), C2 (gérée par M2), C3 (gérée par M3). Si MH2 quitte C2 avant de recevoir m et entre dans C3 après son émission, il ne le reçoit pas. Ainsi, si MH3 reçoit m dans C3, et entre dans M2 avant que le message soit émis, il le reçoit deux fois (figure 06).

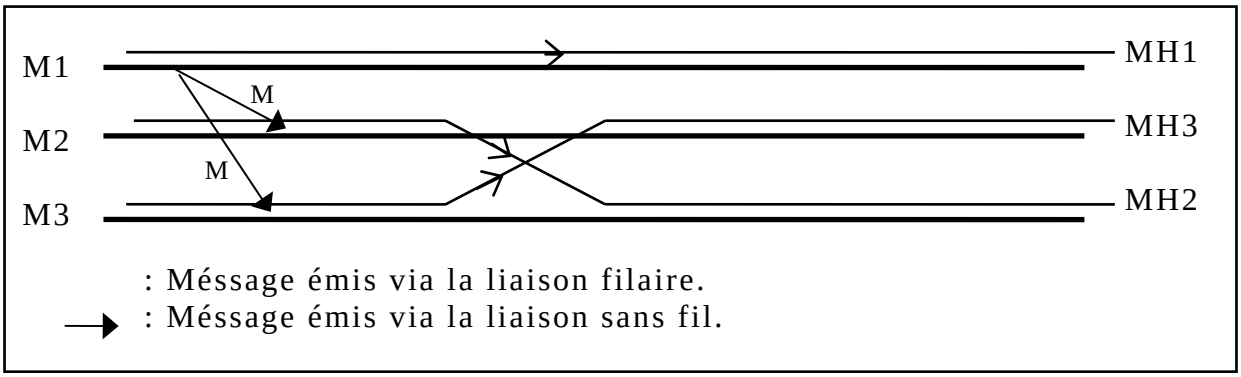


Figure 06: La communication sans fil et la mobilité [Bad, 93]

III.2.2.Gestion des données

La mobilité des sites en dehors de leur serveur habituel peut conduire à des requêtes qui n'étaient pas connues dans les anciens réseaux. Ces nouvelles requêtes nécessiteront de nouveaux mécanismes pour leur gestion parce qu'elles seront dépendantes de la localité des mobiles, telles que l'accès à un serveur de météo ou une requête du genre « Quelle est l'hôpital le plus proche en allant vers le nord ».

Les déconnexions fréquentes des mobiles doivent être prises en compte pour ne pas laisser le système ou la base de données dans un état inconsistant. On peut surmonter les problèmes de déconnexions par l'introduction des mécanismes de *prefetching* (préparation anticipée des données) et des caches sur les mobiles pour leur assurer une certaine indépendance du réseau.

Dans l'environnement mobile, la duplication des données est indispensable pour garantir une bonne disponibilité des données, afin d'augmenter la qualité du service offert aux mobiles.

Il existe une asymétrie dans la communication entre un mobile et une entité fixe qu'il faut remarquer et en profiter. L'accès d'un mobile à une information qui se trouve dans une station fixe ne nécessite pas une localisation de cette station puisque son adresse est fixe. Par contre, l'accès

d'une station fixe à une information située dans un mobile nécessite la localisation de ce mobile. Cette asymétrie aura un très grand impact sur la duplication des données.

Soient deux mobiles qui se déplacent dans leur domaine habituel: un client C qui veut accéder à une information X sur un autre mobile serveur S, donc C doit avoir une copie CX de X. Si cette copie est placée sur le mobile C alors sa mise à jour par S nécessite la localisation de C, mais dans le cas où CX est placée sur une station fixe (serveur de localité de C:LC, ou serveur de localité de S: LS) son accès par C et S ne nécessite pas une localisation (Figure 07).

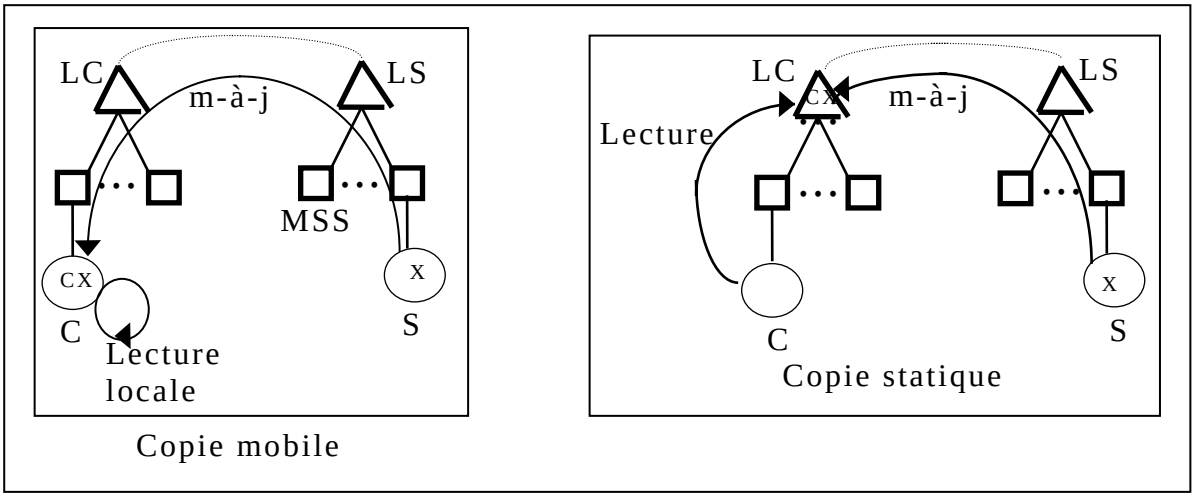


Figure 07: La duplication dans l'environnement mobile

III.2.3.Authentification

Le réseau mobile peut être exposé aux opérations d'usurpation d'identité par les mobiles qui ne lui appartiennent pas. Donc les protocoles d'authentification sont indispensables pour y remédier.

L'authentification d'un mobile qui communique continuellement avec le réseau est simple, car une fois qu'il y a des informations, reçues d'une MSS voisine concernant ce mobile, il est authentifié. Dans le cas où le mobile se déconnecte et se reconnecte sur un autre sous-réseau après un certain temps, son serveur habituel est appelé pour son authentification.

L'augmentation de la sécurité du réseau fixe tout entier est très importante, car si une MSS est piratée, alors tous les mobiles sous cette MSS sont aussi en danger de piratage. Puisque, les messages qu'ils envoient ou qui leurs sont destinés sont piratés aussi. Une solution consiste à éviter de transmettre les informations confidentielles des utilisateurs (mots de passe, clé, ...) entre les sous-réseaux.

Les dispositions suivantes seront nécessaires pour réaliser une bonne authentification [Bag, 95]:

-Eviter de transmettre des informations confidentielles (mots de passe, clés,...) du réseau habituel à des réseaux étrangers.

-L'authentification dans les domaines étrangers doit trouver un impact minimum sur l'interface utilisateur.

-Il faut garder secrets les mouvements des utilisateurs.

-Minimiser le nombre de messages échangés entre le réseau habituel et le réseau étrangers pour identifier un utilisateur.

III.2.4.Securité et confidentialité des données

Généralement, la liaison sans fil est plus exposée à l'espionnage que la liaison filaire, d'où la nécessité des méthodes de cryptage lors de la transmission des informations, que ce soit via la connexion sans fil (d'une MSS à un mobile), ou via le réseau statique (entre le serveur habituel et la la MSS gérant le mobile). Car les données échangées entre un mobile et son serveur habituel ne doivent pas être connues par la MSS couvrant la cellule où évolue le mobile, et les clés de cryptage ne sont détenues que par le mobile et son serveur.

Il est à noter que les méthodes de cryptage et de compression choisies, ou conçues, doivent respecter les contraintes d'énergie sur le mobile et ne doivent pas influer sur la qualité du service offert (transparence pour l'utilisateur) c-à-d choisir des méthodes efficaces et rapides et qui ne demandent pas beaucoup de calcul et d'espace mémoire.

III.3.Les entités mobiles

Vu la limitation de la source d'énergie des mobiles, sa consommation par les différentes opérations de calculs, de stockage et de communication devra être étudiée et réduite au maximum. Pour cela les différents modes sous

lesquels les mobiles peuvent fonctionner peuvent être exploités (mode connecté, en mode veille,...etc).

Ces caractéristiques obligent, lors de la conception des différents algorithmes et protocoles pour l'environnement mobile, de faire la distinction entre ceux qui gèrent les entités fixe et ceux qui gèrent les entités mobiles. Il faut, donc, concevoir pour les mobiles des protocoles plus simples, qui utilisent le minimum de calcul et d'espace mémoire et qui prennent en compte les modes d'opération des mobiles.

IV.Caractéristiques des algorithmes de l'environnement mobile

Les algorithmes distribués actuels conçus pour résoudre les différents problèmes des systèmes distribués, ne respectent pas les nouvelles contraintes de l'environnement mobile, car celles-ci n'étaient pas prises en compte lors de leur conception.

L'application directe de ces protocoles et algorithmes sur les mobiles, sans prendre en compte les spécificités de la connexion sans fil risquent de dégrader les services attendus d'un tel environnement, voir les empêcher.

Prenons l'exemple classique d'exclusion mutuelle, pour accéder à une région critique, utilisant le principe du jeton circulant en anneau logique dans un réseau distribué.

Chaque participant exécute les actions suivantes:

-Attendre le jeton de son prédécesseur dans l'anneau.

-Accéder à la région critique si désiré.

-Envoyer le jeton à son successeur dans l'anneau.

L'application directe de ce protocole entre des participants mobiles se traduit comme suit:

-Attendre le jeton de son prédécesseur dans l'anneau.

-Accéder à la région critique si désiré.

-Rechercher son successeur dans l'anneau.

-Envoyer le jeton à son successeur dans l'anneau.

On remarque plusieurs inconvénients dans un tel protocole:

-Il épuise la source d'énergie des mobiles, car il oblige ceux qui ne sont pas concernés par la région critique, à envoyer et à recevoir le jeton pour permettre aux autres mobiles de l'avoir.

-Il empêche les participants mobiles de se mettre en mode veille.

-coût de localisation très élevé et parfois inutile, par exemples rechercher un mobile qui n'est pas concerné par le jeton.

-Reconfiguration fréquente de l'anneau, provoquée par mouvements, les déconnexions et les reconnections des mobiles.

Pour ces raisons, il sera indispensable de revoir et reconstituer les algorithmes classiques des systèmes distribués tels que l'exclusion mutuelle, la diffusion, la terminaison, l'élection, ... etc, pour assurer leur compatibilité avec les capacités limitées disponibles sur les machines mobiles.

Aprés tous ce qu'on a vu on peut déduire les caractéristiques suivantes que doit respecter un algorithme pour pouvoir s'exécuter sous un environnement mobile:

IV.1.Contraintes sur l'énergie

Comme on l'a vu précédemment, la portabilité (exigée par la mobilité) d'une machine est réalisée au détriment de sa ressource énergétique. La conservation d'énergie est, donc, indispensable pour élargir la durée de vie de la nouvelle source (batterie).

Pour réaliser cela, les algorithmes doivent respecter les contraintes suivantes:

-Minimiser le nombre de messages échangés via la liaison sans fil car cette transmission épuise la source d'énergie.

-Eviter d'obliger le mobile à travailler en mode connecté, et lui laisser lapossibilité de se mettre en mode veille.

-Favoriser la réception sur l'émission puisque cette dernière consomme dix fois plus d'énergie[Bag, 95].

-Essayer d'exécuter la majorité de l'algorithme sur le réseau fixe et de décharger au maximum le mobile de cette tâche.

IV.2.Contraintes sur le médium de communication

Les hôtes mobiles sont souvent déconnectés du réseau, et dans ce cas, ils ne peuvent ni émettre ni recevoir des messages. Un mobile qui se déconnecte au cours de l'exécution d'un algorithme, produit une suspension de cette exécution jusqu'à sa reconnexion. Cette déconnexion est différente d'une panne et un protocole doit s'exécuter avant qu'elle survienne. Une telle possibilité implique que le mobile doit récupérer toutes les données nécessaires pour continuer à travailler indépendamment du réseau (*stand-allone mode*), ou envoyer les données nécessaires pour terminer l'exécution de l'algorithme sur le réseau fixe.

En réalité, un algorithme distribué est exécuté, généralement, par un sous ensemble de mobiles derrière un sous ensemble de MSS qui change en fonction de leurs mouvements. Dans le cas de la panne d'une MSS, cette dernière est immédiatement supprimée de l'ensemble des participants, mais dans le cas où une cellule ne contient plus de mobile, la suppression de sa MSS peut être faite ultérieurement, et elle peut continuer à participer à l'exécution de l'algorithme. L'ensemble des mobiles participants peut changer, aussi, à cause des déconnexions et reconnexions de ses éléments, ce qui est totalement différent des cas de pannes des sites fixes puisqu'un mobile ne peut pas continuer à participer à l'exécution de l'algorithme aprés sa déconnection.

Il faudra, aussi, éviter les algorithmes nécessitants la participation de tous les mobiles. Pour leur laisser la possibilité de fonctionner en mode veille (exemple précédent d'exclusion mutuelle).

Les algorithmes de l'environnemnt mobiles doivent, donc, s'adapter aux contraintes imposées par la connexion sans fil qui sont la limitation de la bande de fréquence et les déconnexions fréquentes. Pour cela, il est envisageable de:

-Réduire au maximum la taille d'un message échangé via le médium sans fil pour minimiser son utilisation.

-Prendre en compte que les déconnexions pouvant se produire à n'importe quel moment et par conséquent éviter de garder les informations

nécessaires pour le déroulement d'un algorithme sur une MSS dans un mobile et vis-versa.

-Introduire le coût de localisation dans l'évaluation des algorithmes mobiles.

-Prendre en compte la possibilité de concentration d'un nombre élevé de mobiles dans une seule région ce qui peut conduire à des problèmes d'engorgement et de dégradation du service.

V.CONCLUSION

L'introduction des hôtes mobiles dans les réseaux répartis actuels nécessitera la revue des algorithmes classiques tel que : l'exclusion mutuelle, l'élection, la terminaison...etc.

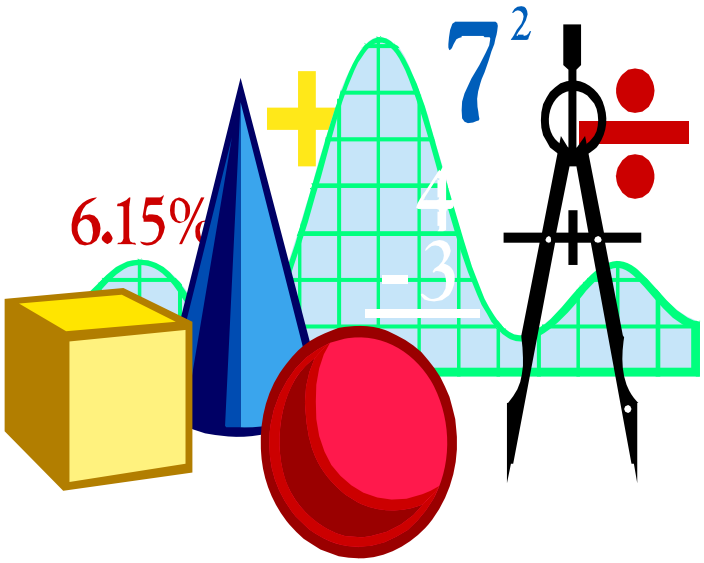
De nouveau paramètres sont à prendre en compte :

- La configuration du réseau n'étant plus fixe à cause de la mobilité des hôtes et leurs déconnexions fréquentes.
- L'énergie disponible sur le mobile est limitée.
- La puissance de calcul et de stockage est faible.
- La bande passante des médiums sans fil est limitée.
- L'introduction des coûts de recherche et de localisation dans l'évaluation d'un algorithme.
- Diffusion naturelle des médiums sans fil.

Dans ce qui suit, on va étudier et comparer des algorithmes distribués en se limitant au problème de l'exclusion mutuelle. Pour essayer d'en extraire le plus adéquat à l'informatique mobile.

CHAPITRE II

ETUDE DES ALGORITHMES DISTRIBUES D'EXCLUSION MUTUELLE



CHAPITRE 2: ETUDE DES ALGORITHMES DISTRIBUES D'EXCLUSION MUTUELLE

I.INTRODUCTION

Depuis plus de trente ans, le phénomène d'exclusion mutuelle constitue l'un des plus beaux paradigmes des difficultés rencontrées dans la programmation parallèle ou distribuée.

Il est important de préciser que la mise en oeuvre d'un mécanisme d'exclusion mutuelle est un phénomène bien réel auquel se trouve confronté tout concepteur de systèmes d'exploitation. Actuellement, les programmeurs d'applications, eux même, se trouvent concernés par ce problème. Parcequ'ils sont, de plus en plus, appelés à utiliser les services offerts par des systèmes informatiques bâtis autour de plusieurs unités de traitements, voire plusieurs ordinateurs réunis au sein d'un réseau.

L'énoncé du problème est simple:

Considérons un ensemble de n processus reliés par un réseau complètement maillé et fiable. Il s'agit d'offrir à chacun de ces processus un protocole lui permettant d'entrer et de sortir d'une section critique c-à-d accessible par un seul processus à la fois. Le protocole doit être exempt d'interblocage, de famine et garantir un ordre de satisfaction entre les requêtes d'entrée en section critique des processus candidats.

Dans ce chapitre, on va faire un survol des systèmes distribués et des techniques qui y sont utilisées. On s'intéressera , en particulier, au problème de l'exclusion mutuelle et aux algorithmes le concernant dans ce modèle.

Enfin, on étudiera les possibilités d'adaptation de ces algorithmes à un environnement mobile.

II.GENERALITES

II.1.Systèmes distribués

Un système réparti informatique se compose d'un ensemble d'ordinateurs et de périphériques reliés entre eux par un système de communication qui leur permet d'échanger de l'information. Un tel système présente une différence essentielle par rapport à un système centralisé: les entités le constituants coopèrent à la réalisation d'un but commun seulement par l'échange de messages. Cela se traduit, d'une part, par l'absence d'une mémoire commune qui sert comme un lieu d'échange d'information, c-à-d, l'impossibilité de définir un état global du système que pourrait obtenir instantanément ses entités. Et d'une autre part par l'absence de référence temporelle (horloge) commune qui peut définir un ordre global des événements qui se produisent dans les différents sites. Un processus ne peut donc percevoir que les événements dont il est le siège et ceux issus des relations de causalité entre l'émission et la réception du même message. D'où l'impossibilité d'ordonner deux événements qui se produisent dans deux sites différents.

Le système de communication reliant les différentes entités à une très grande importance dans les systèmes repartis, et représente l'une de leurs composantes principales. Ces systèmes sont souvent organisés sous forme de couches, réalisant chacune un certain niveau d'abstraction en permettant la communication entre les entités définies à ce niveau.

L'opération élémentaire de communication est représentée par l'émission d'un ensemble d'informations appelé message, d'un site émetteur à un autre destinataire. Les primitives de base d'un système de communication sont donc:

Envoyer (message , destinataire)

Recevoir (message , émetteur)

En plus de ces primitives, un système de communication fournit d'autres fonctions telles que l'établissement et la coupure d'une liaison permanente entre deux sites, qui permet d'acheminer un flot de messages, et le transcodage de l'information.

II.2.Algorithmes distribués

Un algorithme distribué est un ensemble de processus qui coopèrent par l'échange de messages, grâce à un système de communication, pour réaliser un but commun.

La construction des algorithmes distribués fait donc intervenir deux niveaux fondamentaux:

-Niveaux sites: concerne les processus qui reçoivent, émettent et manipulent les données.

-Niveau communication: concerne les liaisons sur lesquelles circulent ces données, c-à-d, la manière de regrouper les différents éléments du système repartis.

II.2.1.Liaisons de communication

En réalité, il existe un certain nombre de topologies qui sont les plus utilisées:

II.2.1.1.Maillage en anneau

Dans cette topologie, un processus ne peut communiquer qu'avec deux autres: son prédécesseur et son successeur. Ces processus forment donc un anneau qui peut être uni ou bidirectionnel.

Cette architecture impose, donc, à un site qui veut communiquer avec un autre qui n'est pas son voisin, de passer par un certains nombre d'autres sites qui ne sont pas concernés par cette communication. ces sites jouent le rôle de noeuds que doit traverser un message pour arriver à son destinataire.

Il est évident que la panne d'un site de cet anneau nécessite sa reconfiguration de façon à ce que cette panne n'isole pas les autres sites.

II.2.1.2.Maillage en étoile

Dans cette structure, toutes les communications passent par un site central qui communique avec tous les autres sites, qui ne peuvent communiquer directement entre eux qu'avec lui. Cette topologie présente un inconvénient majeur lors de la panne du site central.

En effet, La panne du site central dans cette topologie isole complètement les autres sites et détruit toutes liaison entre les entité du réseau.

II.2.1.3.Maillage en arbre

Dans cette topologie, les sites sont organisés sous la forme d'arbre binaire ou m-aire, et un site ne peut communiquer qu'avec son père et chacun de ses fils.

II.2.1.4.Maillage complet

Chaque site peut communiquer directement avec n'importe quel autre site.

Les algorithmes distribués utilisent l'une de ces topologies ou l'autre, mais dans le cas où l'algorithme nécessite un maillage spécifique différent de la topologie physique, il sera nécessaire d'ajouter une couche supplémentaire pour simuler ce maillage.

II.2.2.Hypothèses sur les liaisons de communication

En plus de la topologie, les algorithmes distribués font toujours des hypothèses sur les liaisons de communication telles que:

-La non-duplication des messages: Un message n'est émis et reçu qu'une fois au plus.

-La non-altération des messages: Un message émis est reçusans érreur et sans perte.

-L'ordre d'émission et le même que celui de la réception.

-Le délai d'acheminement d'un message est fini.

Un algorithme, basé sur une hypothèse qui n'est pas disponible sur la voie de communication, nécessite l'ajout d'une couche logicielle réalisant le transport doté des propriétés voulues.

II.2.3.Evaluation des algorithmes distribués

D'après ces définitions, divers critères peuvent être tirés et être utilisés comme paramètres pour l'évaluation et la comparaison des algorithmes distribués:

II.2.3.1.Degré de distribution

Ce critère concerne la manière dont l'algorithme est reparti sur les différents sites, c-à-d, la symétrie des rôles joués par les différents processus:

-Non-symétrie: chaque processus exécute un texte différent.

-Symétrie de texte: chaque processus exécute le même texte que les autres, mais cette exécution est liée à des paramètres locaux à ce processus.

-Symétrie forte: tous les processus exécutent le même texte indépendamment de leurs paramètres, mais selon le cas dans chaque processus (messages reçus).

-Symétrie totale: tous les processus exécutent le même texte et se comportent de manières identiques.

II.2.3.2.Résistance aux pannes

Plus un site joue un rôle important dans l'exécution d'un algorithme, plus sa panne présente un danger pour cette exécution (blocage, état inconsistant,...). Les algorithmes qui présentent des degrés de répartition assez élevés peuvent continuer à s'exécuter (en mode dégradé) même aprés la panne d'un certain nombre de sites.

II.2.3.3.Hypothèses sur le réseau

Les algorithmes les plus adaptés sont ceux qui font moins d'hypothèses sur les voies de communication et ceux dont l'exécution ne nécessite pas un environnement précis.

II.2.3.4.Complexité

Le nombre de messages échangés dans le réseau et la charge induite au niveau de ce dernier par l'exécution d'un algorithme, ainsi que les temps d'attente des processus qui y participent sont des paramètres qui interviennent fortement dans l'évaluation et la comparaison des algorithmes distribués

II.2.3.5.Etat global ou local

Les algorithmes qui s'adaptent mieux aux systèmes distribués sont ceux qui peuvent prendre des décisions en se basant sur leur état local et sans avoir besoins de connaître l'état global du système. Ce critère permet de minimiser le nombre de messages et de résister aux pannes éventuelles des sites.

II.3.Techniques de l'algorithmique distribué

Les algorithmes distribués utilisent, généralement, un certain nombre de techniques pouvant être utilisées chacune pour résoudre des problèmes particuliers:

II.3.1.Calcul diffusant

Dans cette technique, peu importe la topologie utilisée. Un processus P_0 joue un rôle particulier, il peut initialement émettre des messages alors que les autres ne le peuvent pas tant qu'ils n'ont pas reçu un message.

Le calcul diffusant peut être utilisé pour réaliser des contrôles particuliers tel que la terminaison.

II.3.2.Le jeton circulant

C'est une technique très simple qui consiste à faire circuler un privilège entre un ensemble de processus connectés en structure d'anneau. L'anneau peut être défini de manière statique ou reconfiguré dynamiquement.

II.3.3.L'estampillage

Ce principe, appelé aussi horloge logique, est proposé par *Lamport* en 1978. Il sert à ordonner les événements qui se produisent dans un algorithme distribué.

Chaque processus P_i gère un compteur local h_i appelé horloge de P_i, et chaque message émis par P_i est marqué par son horloge h_i, les messages ont donc la forme:

message: (m, h_i, i)

Lorsque P_i reçoit un message (m, h_i, i) de j , son horloge aura la valeur [max (h_i, h_j) + 1], et sera considérée comme la date de réception de ce message.

Lorsque P_i émet un message (m, h_i, i) à P_j, il incrémente son horloge hi avant de l'inclure dans le message.

Un message (m_1,h_i,i) est considéré antérieur à un message (m_2,h_j,j) si ($h_i < h_j$) ou ($h_i = h_j$ et $i < j$).

Dans les algorithmes d'exclusion mutuelle, cette technique est utilisée généralement pour ordonner une file d'attente distribuée dans les différents sites.

II.4.Exclusion mutuelle

Le problème d'exclusion mutuelle est l'un des premiers problèmes apparus avec la programmation parallèle. Ce phénomène, essentiel pour la conception des systèmes d'exploitation, consiste à établir des opérations de base permettants de résoudre le conflit résultant du partage de ressources dans un système informatique par un ensemble de processus. Ce problème peut être posé, donc, lors de l'accès à un fichier, un mot mémoire, une imprimante,... etc. L'exemple suivant montre l'importance du problème:

Soient deux processus $P_{virement}$ et Q_{paye} qui veulent modifier une variable commune C (état d'un compte bancaire).

$P_{virement}$ execute : $C := C + C_{virement}$	$Q_{retrait}$ execute: $C := C + C_{retrait}$
c-à-d lire(C)	lire(C)
$C := C + C_{virement}$	$C := C - C_{retrait}$
ecrire(C)	ecrire(C)

Si Q (resp.P) lit C avant qu'elle soit écrite par P (resp.Q), la valeur finale de C sera $C - C_{retrait}$ (resp.$C := C + C_{virement}$) au lieu de $C + C_{virement} - C_{retrait}$.

Pour éviter cette incohérence, les instructions de P et Q doivent être exécutées en exclusion mutuelle, c-a-d, que leurs exécutions doivent être indivisibles (atomique).

Au niveau de la programmation, l'ensemble des instructions, relatives à l'utilisation de la section critique, sont appelées *sections critiques*.

Pour contrôler l'accès à une ressource, un protocole formé de deux parties qui sont demande de la ressource et sa libération est utilisé. La première précédant cet accés et la deuxième le succédant, comme suit:

<Demande d'entrer en section critique>

<Accès à la section critique>

<Libération de la section critique>

Pour qu'il soit opérationnel un algorithme d'exclusion mutuelle doit avoir au minimum les propriétés suivantes dues à DJIKSTRA:

a.A tout instant, un processus au plus peut se trouver en section critique.

b.Si plusieurs processus sont bloqués en attente de la section critique, alors qu'aucun ne s'y trouve, l'un d'eux doit y accéder au bout d'un temps fini.

c.Si un processus est bloqué hors d'une section critique, son blocage ne doit pas empêcher un autre processus d'y accéder.

d.La solution doit être la même pour tous les processus et aucun d'eux ne joue un rôle privilégié.

Dans un contexte distribué, les algorithmes d'exclusion mutuelle ne reposent plus sur l'accès à une mémoire commune, mais plutôt sur l'échange de messages, ils doivent donc être exempts d'interblocage et équitables.

III.Algorithmes distribués d'exclusion mutuelle

Les algorithmes résolants le problème de l'exclusion mutuelle ne comportent, en général, pas plus d'une dizaine de lignes de programmes et ne contiennent que de triviales instructions d'affectation. Pourtant, le parallélisme rend difficile la compréhension de leurs comportements et l'analyse de leurs propriétés comme l'absence de blocage ou l'arbitrage équitable des conflits.

Dans un environnement distribué, les algorithmes ne reposent sur aucun dispositif centralisé tel que mot mémoire ou horloge. Sous cette définition, sont regroupés deux classes d'algorithmes distribués qui se différencient par leur mode d'expression et les mécanismes sous adjacents qu'ils supposent.

Il existe, en effet, deux modes d'expression des algorithmes distribués d'exclusion mutuelle. Le premier utilise des variables d'état, et le second utilise la communication de messages pour véhiculer les informations nécessaires entre les processus. Dans ce qui suit, on va présenter les algorithmes de chacun de ces types sans s'intéresser à leurs preuves qui sont discutés dans différents ouvrages (voir[Ray, 84]).

III.1.Algorithmes basés sur les variables d'état

Cette méthode est basée sur l'utilisation de variables d'état. Ces variables sont locales et propres à chaque processus, si l'on se réfère à la terminologie utilisée dans [Ray, 84] qui dit qu'une variable est locale à un processus si elle est inaccessible aux autres processus et qu'une variable est propre à un processus si celui-ci est le seul à pouvoir à la fois la lire et l'écrire alors que tous les autres ne peuvent que la lire.

Donc, en aucun cas, un processus ne peut influencer de façon directe l'état d'un autre processus (l'état d'un processus est représenté par ses variables d'état), mais il pourra quand même le connaître (la lecture d'une variable se traduit par une demande envoyée à son propriétaire, qui répond en envoyant la valeur de cette variable): Chaque processus est autonome et aucun dispositif centralisé n'est utilisé pour régler les conflits d'accès à la section critique.

L'inconvénient majeur de ce type d'algorithmes est le nombre très élevé de messages nécessaires pour gérer l'accès à la section critique. Il faudrait noter que les processus testent eux mêmes les variables d'état qui les intéressent de façon permanente jusqu'à la satisfaction de leurs conditions d'attente. Ces tests aveugles, et souvent inutiles, augmentent énormément le nombre de messages échangés via le réseau.

Il semble, donc, que ces algorithmes peuvent être utilisés dans des systèmes centralisés (de part l'indépendance remarquable qu'ils offrent aux différents processus) plutôt que dans des systèmes distribués (de part le

nombre élevé de messages qu'ils utilisent). La résistance aux pannes de ces algorithmes est généralement bonne: la panne d'un processus n'affecte pas le comportement global de l'algorithme. Dans ce qui suit, on va citer les principes et les propriétés des différents algorithmes d'exclusion mutuelle de cette classe sans donner leurs détailles que le lecteur pourra consulter dans l'annexe.

III.1.1.Algorithme de la boulangerie [Lamport 74]

Cet algorithme est basé sur l'idée utilisé habituellement dans les magasins, où chaque client reçoit à l'entrée un ticket contenant le numéro de son tour. Dans l'algorithme, un processus choisis son numéro en fonction des numéros pris par les autres. Si deux processus choisissent le même numéro, la priorité sera attribuée à celui dont l'indice est inférieur.

Cet algorithme, basé sur une symétrie de texte et sur un maillage complet, se classe dans une classe d'algorithmes utilisants la technique d'estampillage pour ordonner les différentes requêtes. Chaque procéssus execute les instructions suivantes:

-Choisir un numéro supérieurs à tout les numéros choisis (se déclarer candidat pour l'accés à la section critique).

-Attendre que ce numéro soit le plus grand des numéros choisis.

-Accéder a la section critique.

-Mettre son numéro à 0 (se déclarer non candidat).

Il est à noter que l'algorithme pose le problème de la croissance infinie de la variable numéro[i].

III.1.2.Algorithme auto-stabilisateur [DJIKSTRA 74]

Cet algorithme fait circuler un privilège entre les différents processus qui sont disposés selon une structure d'anneau où un processus P_i ne peut demander des informations qu'à son voisin de droite $P_{(i-1 \bmod n)}$.

Le processus P_0 joue un rôle particulier qui est l'initialisation du privilège.

Cet algorithme présente l'inconvénient d'obliger un processus à prendre le privilège même s'il ne désire pas entrer en section critique. Ce qui augmente le nombre de messages nécessaires pour dérouler l'algorithme.

La panne d'un site dans cet algorithme nécessite la reconfiguration de l'anneau c-à-d avertir les voisins du site en panne pour qu'ils changent les informations le concernant.

III.1.3.Algorithme de Hehner-Symasunder

C'est un algorithme similaire à celui de la boulangerie de Lamport, vu précédemment, mais il utilise une seule variable.

Dans cet algorithme tous les processus suivent le même protocole pour pénétrer dans la section critique. A chaque processus est associée une variable qu'il est le seul à pouvoir lire et écrire, les autres peuvent seulement la lire.

Ce protocole, utilisant la technique d'estampillage, définit un comportement non symétrique pour les processus.

III.2.Algorithmes basés sur la communication de messages

Un autre type d'algorithmes très utilisé dans les systèmes distribués est celui basé sur la communication de messages. Les processus ne demandent plus les informations des autres, mais ils en reçoivent. Chaque fois qu'un processus change d'état, il diffuse cette modification à tous les autres. Ce principe permet de réduire le nombre de messages échangés entre les processus, en éliminant les messages de test inutiles dans le cas où les variables testées ne changent pas d'état. Ces algorithmes présentent l'avantage d'être indépendants de la technologie et de la topologie du réseau où ils s'exécutent, grâce au mode d'expression qu'ils utilisent (communication de messages).

Ces algorithmes reposent, généralement, sur le principe de l'estampillage pour gérer les messages transmis.

L'exécution de ces algorithmes nécessite l'utilisation de deux processus sur chaque site, le premier demande l'accès à la section critique et le deuxième manipule les messages reçus des autres sites et met à jour les variables utilisées par le premier. Ces variables représentent donc des sections critiques pour ces deux processus et leur gestion peut être faite en utilisant les algorithmes centralisés d'exclusion mutuelle.

III.2.1.Algorithme de Lamport[78]:Distribution d'une file d'attente

C'est l'un des algorithmes distribués d'exclusion mutuelle utilisant le principe de l'horloge logique, vu précédemment, pour ordonner les demandes d'entrées en section critique des différents processus. Il est caractéristique d'une classe d'algorithmes qui reposent sur l'utilisation d'une file d'attente distribuée, dont il existe une représentation partielle sur chaque site.

Principe: Un processus, qui veut entrer en section critique, transmet une demande datée de son horloge à tous les autres processus. Un processus n'entre en section critique que si sa demande est plus ancienne (horloge logique) que toutes les autres demandes, et qu'il a reçu un accord de tous les autres processus, pour garantir qu'un message antérieur à tous les autres n'est pas en transit.

Protocole d'accès à la SC:

-Diffuser une requête datée de son horloge locale.

-Attendre que sa requête soit la plus ancienne.

-Entrer en section critique.

-Diffuser une libération de la section critique.

Cet algorithme garantie l'exclusion mutuelle, est équitable et est exempt d'interblocage et peut être amélioré pour résister aux pannes en ajoutant un champ absent dans la file pour chaque site, ce champ est modifié en fonction d'un message (absent,k,i) provoqué par le réseau de transport indiquant la panne de ce site. Ce champs sera consulté par chaque processus avant de comparer son horloge avec une autre, si ce champs est vrai alors la comparaison est annulée.

Le nombre de messages nécessaires pour effectuer un accès à la section critique par un processus est 3(n-1):

n-1 messages pour diffuser la demande.

n-1 messages pour recevoir les accords.

n-1 messages pour diffuser la libération.

III.2.2.Algorithme de Ricart & Agrawala

Cet algorithme représente une optimisation de celui de Lamport en éliminant les messages accusés de réception des messages REQUEST. Ainsi, le nombre de messages véhiculés dans le réseau pour accéder à une section critique devient 2(n-1):

n-1 messages pour diffuser la demande.

n-1 messages pour recevoir les réponses à une demandes.

Protocole d'accès:

-Diffuser une requête datée de son horloge.

-Attendre des réponses favorables de tous les autres sites (en répondant, pendant l'attente, aux requêtes plus ancienne et en différant les nouvelles)

-Accéder à la section critique.

-Répondre aux requêtes différées.

Un processus voulant entrer en section critique diffuse une demande, éstampillée de son horloge locale, à tous les autres processus. Il n'y pénètre

que s'il reçoit les réponses favorables de tous les autres processus. Un processus qui reçoit une demande peut se trouver dans les cas suivants:

-il ne veut pas accéder à la section critique, alors il répond immédiatement au processus demandeur.

-il veut y accéder, alors il compare la date de cette demande avec la sienne: si elle est antérieure il répond immédiatement sinon il retarde la réponse jusqu'à sa sortie de la section critique.

L'algorithme suppose un réseau de transport sans erreurs, ou les temps de transite sont variables et les messages peuvent se doubler.

III.2.3.Algorithme de LANN

Une autre classe d'algorithmes basés sur la communication de messages repose sur le principe de circulation d'une information particulière appelée jeton entre un ensemble de sites organisé en anneau virtuel, définis par une numérotation des sites choisis par convention de 0 à n-1 (n étant le nombre de sites constituants l'anneau). Le jeton circule dans le sens des numéros croissants. A l'initialisation du système, un site et un seul possède le jeton. Le jeton représente, donc, l'autorisation d'accès à la section critique.

L'implémentation de ce protocole connaît un certain nombre de problèmes tel que la panne d'un site au moment où il détient le jeton ou la panne du système de communication, ce qui implique la perte du jeton. La résolution de ces problèmes nécessite de nouveau protocoles pour la détection de la perte du jeton et sa régénération.

Un autre problème est posé par la panne d'un processus ce qui nécessite une reconfiguration de l'anneau logique. Généralement cette tâche est à la charge du système de transport qui doit fournir à chaque processus deux variables d'état: voisin gauche, voisin droit: 0..n-1. Et assurer leur mise à jour lors de la panne d'un site.

Le nombre de messages nécessaires pour pouvoir accéder à une section critique, en utilisant cet algorithme, varie de 0 dans le cas où tous ces processus demandent d'accéder à la section critique, à une infinité dans le cas où aucun processus ne désire y accéder.

III.2.4.Algorithme de Susiki & Kasami

C'est une optimisation de l'algorithme de Lann. Il consiste à accompagner le jeton, circulant entre les sites, d'un tableau de n entier contenant des informations sur les dernières requêtes satisfaites des différents sites. Un processus qui demande d'accéder à la section critique diffuse une demande avec un numéro supérieur à celui circulant avec le jeton. Un processus P_i venant de sortir de la région critique envoie le jeton au premier de ses voisins dans l'ordre $P_{i+1},\ldots,P_n,P_0,\ldots,P_{i-1}$ ayant le numéro de sa requête supérieur à celui dans le jeton. Cette technique permet d'éviter l'envoie du jeton à ceux qui ne demandent pas d'entrer en section critique.

L'algorithme permet une optimisation du nombre de messages nécessaires pour une exclusion mutuelle:

-n messages dans le cas où plus d'un processus demandent la ressource.

-0 si le processus demandant la ressource possède déjà le jeton.

L'inconvénient de cet algorithme réside dans la taille du message du jeton. En plus de l'information particulière du jeton, le message est constitué d'un tableau de n éléments.

L'algorithme garantie l'exclusion mutuelle, et si les messages sont délivrés dans un temps fini alors il est exempt d'interblocage.

Dans cet algorithme, le jeton n'est pas une simple information connue par tous les sites, comme dans le cas de l'algorithme de Lann. C'est un ensemble d'informations liées à tous les sites et dépendant de l'évolution des demandes dans ces sites. Et par conséquent les techniques de régénération du jeton après sa perte, utilisées dans l'algorithme de Lann, ne peuvent pas le faire dans cet algorithme.

```
Processus Pi
Var   osn                    :0..+∞;
      JetonPrésent,Dedans    :booléen;
      Jeton,Requête          :tableau[1..n] de 0..+∞;
Protocole d'accès
      Si non(JetonPresent) Alors
            Debut
              osn:=osn+1;
              Diffuser(REQUEST,osn,i);
              Attendre(Jtn,Jeton)
            Fin
      Dedans:=vrai;
      JetonPresent:=vrai;

      <Section critique>
      Jeton[i]:=osn;
      Dedans:=faux;
      Pour j de i+1..n, de 1..i-1 Faire
            Debut
              Si requête[j]>Jeton[j] et JetonPresent Alors
                    Debut
                      JetonPresent:=faux;
                      Envoyer(Jtn,Jeton) à j;
                      Exit;
                    Fin
            Fin.

Gestion des messages:

 A la réception de (REQUEST,k,j) de j Faire
      Debut
        Requête[j]:=max(Requête[j],k);
        Si JetonPresent et non(Dedans) Alors
            Debut
              Pour j de i+1..n, de 1..i-1 Faire
                    Debut
                      Si requête[j]>Jeton[j] et JetonPresent Alors
                            Debut
                              JetonPresent:=faux;
                              Envoyer(Jtn,Jeton) à j;
                              Exit;
                            Fin
                    Fin
            Fin
      Fin.
```

IV.Exclusion mutuelle & environnement mobile

L'un des buts principaux de l'informatique mobile est d'offrir aux utilisateurs les mêmes possibilités et les mêmes services qu'offre les systèmes statiques, c-à-d, surmonter les problèmes induits par la mobilité des utilisateurs. Parmi ces services, le partage des ressources distribuées sur les différents hôtes mobiles ou fixes entre ses hôtes. L'accès à ces ressources tel que les fichiers, les imprimantes et les mémoires, peut, comme dans l'environnement statique, créer des problèmes de cohérence. Le problème qui se pose est :'Est ce que les solutions proposées pour garantir l'accès en exclusion mutuelle à ces ressources dans les systèmes distribués classiques, peuvent le faire dans un environnement mobile'. Pour répondre à une telle question, il faudrait en premier lieu définir le modèle du réseau mobile et spécifier les ressources que doivent partager les utilisateurs du système.

IV.1.Modèle

Le modèle le plus utilisé pour répondre aux besoins de la mobilité, c-à-d, qui offre aux utilisateurs la possibilité de garder la connexion avec le réseau tout en se déplaçant est le modèle cellulaire décrit dans la section[II.5.5] du chapitre précédent.

Dans ce modèle, les mobiles qui sont identifiés par une MSS disposent, dans cette dernière, d'une table concernant les identités des mobiles locaux, ainsi que les informations qui leur sont spécifiques.

Un hôte mobile (MH) peut communiquer directement (via la liaison sans fil) avec une MSS seulement s'il se trouve physiquement dans la cellule couverte par cette MSS. Dans un instant donné, un mobile ne peut appartenir qu'à une seule cellule, qui représente sa position actuelle.

Le réseau statique, supposé fiable, délivre séquentiellement les messages entre les MSSs avec une latence arbitraire.

Le réseau sans fil, dans une cellule, assure une liaison FIFO des messages entre la MSS et les MHs locaux. Tant qu'un mobile n'a pas quitté une MSS, il reçoit séquentiellement les messages qui lui sont envoyés par cette MSS. Dés qu'il doit la quitter, il ne recevra qu'un préfixe de la

séquence envoyée par celle-ci, donc le mobile doit envoyer un message à la nouvelle MSS M' voulant dire: 'Moi le mobile Mh-id venant de quitter M aprés la réception du message numéro r. Je demande d'être reçu par la MSS M' gérant la cellule dans laquelle je me trouve'. Alors, M' ajoute Mh-id à sa liste de mobiles locaux et communique avec M en lui envoyant Let(Mh-id,r) pour transférer les paramètres de Mh-id et les messages qui doivent lui être transmis à partir de message numéro r.

Un mobile, qui se déconnecte, envoie un message Disconnect(r) à sa MSS locale M, où r est le numéro du dernier message reçu de M. Cette dernière le supprime de sa liste des mobiles locaux et met à jour un indicateur pour informer ceux qui le cherche de sa déconnexion.

Quand un mobile veut se reconnecter à une nouvelle MSS, il envoie un message Reconnect(Mh-id) à la MSS courante, qui l'ajoute à sa liste de mobiles locaux et envoie Let(Mh-id) à Ex-MSS qui lui retourne ses paramètres et met à jour l'indicateur de déconnexion.

Dans cet environnement, pour évaluer les algorithmes, de nouvelles mesures sont apparues, tel que le coût d'utilisation du médium sans fil, le coût de recherche d'un mobile dans le réseau. On utilise pour cela les mesures de coût suivantes:

C_{fixe} : Coût induit par l'envoie d'un message via le réseau statique.

$C_{wirelessl}$: Coût induit par l'envoie d'un message via la liaison sans fil (entre un mobile et une MSS).

C_{search}: Coût de localisation d'un mobile. Ce coût dépend de la méthode de localisation utilisée.

IV.2.Exclusion mutuelle sur les canaux de communication

IV.2.1.Position du problème

Dans un réseau mobile, basé sur le modèle cellulaire, pour qu'une communication (ou transfert de données) entre une MSS et un mobile puisse avoir lieu, on a besoin d'établir une liaison sans fil (canal radio) entre ces deux entités. Ce canal ne doit pas être utilisé dans les cellules voisines durant

cette session de communication pour éviter l'interférence entre ces canaux dans les différentes cellules (voir Chap1.II.5.5.1). Il s'agit, donc, de trouver un moyen pour diviser l'ensemble des canaux disponibles entre les cellules.

Puisque le nombre de canaux est limité, on a besoin de distribuer l'ensemble de ces canaux entre les différentes stations de base. Les méthodes statiques qui divisent l'ensemble des canaux entre les MSS une fois pour toutes, ne permettent pas de supporter le changement de la charge (nombre de mobiles) dans les différentes cellules. Cela augmente le temps de latence des mobiles, bien qu'il existe des canaux qui peuvent être utilisés. D'où la nécessité des méthodes dynamiques (voir Chap1.II.5.5.1.2).

Les MSS sont, donc, concurrentes pour l'obtention de ces canaux afin de supporter les sessions de communication dans leur cellules. Cela peut conduire à des conflits lors de l'allocation d'un canal à une MSS ou l'autre, ce qui est similaire à un problème d'exclusion mutuelle. Cependant, le cas d'allocation des canaux de communication est plus général qu'un problème d'exclusion mutuelle classique:

-Une MSS va pouvoir communiquer avec plusieurs mobiles en même temps, en utilisant un canal différent pour chaque session de communication avec un mobile. Cela signifie qu'une MSS peut se trouver dans plusieurs sections critiques en même temps.

-Les algorithmes distribués d'exclusion mutuelle existants supposent qu'un site spécifie l'identité de la ressource à laquelle il veut accéder. Par contre dans le cas d'allocation des canaux, une MSS demande n'importe quel canal non utilisé par ces voisines.

-Les algorithmes connus d'exclusion mutuelle n'imposent aucune restriction sur le temps entre la demande d'une ressource par un site et son attribution à ce dernier. Dans notre cas, ce temps est très important.

pour toutes ces raisons, un nouvel algorithme qui prend en compte ces nouvelles contraintes est nécessaire.

IV.2.2.Solution centralisée

A première vue, une solution dynamique centralisé est envisageable. On charge une station fixe de la gestion de l'ensemble des canaux et de leur affectation. Une station qui veut allouer un canal n'a qu'à le demander de la

station centrale. Cette solution peut conduire a des surcharge dans le réseau fixe si le nombre de sessions de communication augmente, et peut même conduire a l'engorgement de la station centrale. D'où la nécessité d'une méthode dynamique distribuée.

IV.2.3.Solution distribuée

Cette solution est présenté par un algorithme proposé par Singhal et Parakash en 1995 à l'université de Rutgers aux Etats unis [Par,95], dans lequel les canaux de communication sont alloués dynamiquement aux différentes MSSs. La MSS d'une cellule envoie des demandes, estampillées par une horloge de Lamport, à ses voisines immédiates pour déterminer le canal qui peut être utilisé dans une session de communication dans cette cellule. Parfois, un canal doit être transféré d'une MSS à une autre. Pour que ce transfert ne conduit pas à des interférences, l'algorithme exige qu'une MSS n'alloue un tel canal qu'après avoir reçu des réponses positives de toutes ses voisines.

Un canal alloué à une MSS, y reste jusqu'à ce qu'il fasse l'objet d'un nouveau transfert. Ce qui s'adapte bien avec le changement de la charge dans la cellule correspondante. L'algorithme est présenté dans le chapitre suivant aves ses détailles et ses preuves.

IV.3.Exclusion mutuelle classique

Les hôtes mobiles, tout comme ceux fixes, peuvent entrer en concurence pour accéder à des ressources informatiques critiques telles que fichiers, variables, imprimantes,... etc. Pour pouvoir contrôler les conflits résultants d'une telle concurrence, on aura besoin de protocoles d'accès à ces ressources appelés algorithmes d'exclusion mutuelle. La question qui se pose est la suivante: « Est ce que les algorithmes déjà présentés peuvent en même temps gérer ces conflits et respecter les contraintes de l'environnement mobile ». Pour répondre à cette question, on va prendre deux exemples d'application directe de ces algorithmes et voir leurs avantages et leurs inconvénients.

Prenons l'exemple de l'algorithme de Lamport basé sur la communication de messages. L'application directe de cet algorithme dans

l'environnement mobile consiste en son exécution au niveau de chaque mobile. Le nouvel algorithme est le même que celui décrit dans [III.2.1], sauf qu'on précède chaque instruction Envoyer(message, i) par une instruction de recherche du destinataire Chercher(i).

La transmission de chaque message d'un mobile à un autre engendre le coût suivant:

$$2C_{wireless} + C_{fixe} + C_{search}$$

Et puisque l'algorithme classique nécessite la communication de $3(N_{MH}-1)$ messages pour satisfaire une demande, le nouvel algorithme engendrera le coût suivant pour le faire:

$$3(N_{MH} - 1)(2C_{wireless} + C_{fixe} + C_{search})$$

N_{MH}:Nombre de mobiles.

Chaque mobile à besoin d'envoyer $2(N_{MH} - 1)$ messages à sa MSS et d'en recevoir $(N_{MH} - 1)$ pour pouvoir accéder à la région critique. Le coût $(6(N_{MH} - 1)C_{wireless})$ est très élevé. Il épuise d'une part la batterie du mobile émetteur et ceux des récepteurs, et surcharge, d'autre part, la liaison de communication sans fil, qui est faite pour la transmission des données pures, par ces messages de contrôle.

Le coût $3(N_{MH} - 1)(C_{fixe} + C_{search})$ d'acheminements des messages entre les MSSs et de localisation des mobiles surcharge le réseau statique.

Les instructions exécutées sur le mobile telles que la gestion de la file f, consomment aussi de l'énergie.

L'accès d'un mobile à la section critique nécessite le réveil de tous les mobiles qui peuvent opèrer en mode veille pour répondre à sa requête.

Le deuxième exemple de l'application directe est l'algorithme de Lann basé sur le principe de jeton. Cette application donne l'algorithme décrit dans la section (4) du chapitre précédent. Le coût induit par cet algorithme est:

$$N_{MH} (2C_{wireless} + C_{fixe} + C_{search})$$

Et les mêmes inconvénients de l'algorithme précédent sont présents.

Pour remédier à tous ces inconvénients, le principe des Deux-tiers est proposé.

IV.3.1.Principe des Deux-tiers

Ce principe est ainsi appelé parcequ'il propose que les deux tiers d'un algorithme pour l'environnement mobile soient exécutés sur la partie statique du réseau (sur les MSSs) [Bad, 93]. Cela permet de décharger les mobiles des deux tiers du coût d'une application distribuée, et libérer le médium sans fil de la plupart des messages de contrôle liés à cette application.

Appliquons ce principe sur l'algorithme de Lamport déjà présenté: La file d'attente f est distribuée dans toutes les MSSs, elle contient toutes les demandes d'accès à la séction critique du réseau, qui sont envoyées par les mobiles à leur MSS locale. On n'a pas donc à chercher un mobile pour lui envoyer l'autorisation d'accès. Dès que son tour arrive l'autorisation lui est envoyée par la MSS de la cellule où il se trouve. Une MSS, qui reçoit une demande d'accès d'un mobile, communique avec toutes les autres pour définir la plus basse priorité à donner à cette demande. On aboutit à l'algorithme suivant:

♦ Actions exécutées par un mobile Mhi

□Pour demander la ressource:

- Envoyer (DEMANDE, MHi) à la MSS courante.

□A la réception de (ACCES, MSSi) de MSSi:

- Section critique
- Envoyer (LIBERER, MHi) à MSSi

♦ Action exécutées par une MSS M:

Var f : File d'attente de (identite-mobile,horloge,servi:boolean) init ∅;

/* Ordonnée selon l'ordre croissant du champ horloge. Les élément ayant servi à faux sont les moins prioritaires */

□A la réception de (DEMANDE, MHi) de Mhi

- Enfiler(f,MHi,false);
- Diffuser(REQUEST,MHi) aux MSSs;
- Attendre la réception de (REP,MHi,kj) de toutes les MSS Mj;
  - -f(MHi).horloge:=Max(kj reçus)+1;
  - -f(MHi).servi:=true;
- Diffuser(SERVIR,MHi,f(MHi).horloge) aux MSSs;

□A la réception de (REQUEST,MHj) de M'

- Enfiler(f,MHj,false);
- Envoyer (REP,Max(f.horloge)+1,MHj) à M';

□A la réception de (SERVIR,MHj,k) de M'

- file(MHj).horloge:=k;
- file(MHj).servi:=true;

□A la réception de (LIBERER, MHi) de Mhi

- Défiler(f,MHi);
- Diffuser(RELEASE,MHi) aux MSSs;

□A la réception de (RELEASE,MHj) de M'

- Défiler(f,MHj);

□ Une MSS M envoie (ACCES,M) à un mobile MHi si et seulement si:

- MHi est local à M.
- MHi est à la tête de f.
- f(MHi).servi=true.

$$3C_{wireless} + 4(N_{MSS} - 1)C_{fixe}$$

$3C_{wireless}$: Coût d'envoi d'une demande par un mobile + Le coût de réception de l'autorisation d'accès + Le coût d'envoie du message de libération.

$4(N_{MSS} - 1)C_{fixe}$: Coût de diffusion de la requête dans le réseau fixe + coût de réception des dates maximales + coût de diffusion de la date effective + coût de diffusion de l'annulation de la requête.

Cet algorithme :

-Décharge les mobiles de l'envoie et de la réception de $(6(N_{MH}-1)-3))$ messages via la liaison sans fil. Ce qui se traduit par une diminution de l'énergie consommée sur le mobile et une libération du médium sans fil.

-Décharge les mobiles des opérations (CPU, mémoire, ...) de gestion des files d'attente.

-Diminue le nombre de messages circulants dans le réseau statique car $(N_{MSS} << N_{MH})$

-Ne nécessite aucun coût de localisation des mobiles parce que les informations concernant leurs demandes sont dans toutes les MSSs.

L'application du principe des deux tiers sur l'algorithme de Lann consiste à faire circuler le jeton entre les MSSs au lieu des mobiles. Chaque mobile n'a qu'à envoyer une demande à sa MSS courante. Lorsque le jeton atteint cette MSS, elle recherche le mobile pour lui délivrer une autorisation d'accès en section critique. L'algorithme est le suivant:

Algorithme:

♦ Actions exécutées par un mobile Mhi □Pour accéder à la région critique •Envoyer (DEMANDE,Mhi) à la MSS courante. □A la réception du (ACCORD)de M
<Section critique>
•Envoyer (LIBERER,Mhi) à M;

♦ Actions exécutées par une MSS Mi

Var file-servie, file-demande: file;

□A la réception de (DEMANDE,Mh)

- Enfiler(file-demande,Mh)

□A la réception du jeton de son prédécesseur Mi-1

- file-servie:= file-demande;
- file-demande:= vide;
- Si non(filevide(file-servie)) Alors

```
Debut
  Défiler(file-servie,Mh);
  Chercher(Mh);
  Envoyer(ACCORD,M) à Mh;
End
```

□A la réception de (LIBERER de Mh)

- Si non(filevide(file-servie)) Alors

```
Debut
  Défiler(file-servie,Mh);
  Chercher(Mh);
  Envoyer(ACCORD,M) à Mh;
End
Sinon
  Envoyer (jeton) à son successeur dans l'anneau;
```

Le coût est donc:

$N_{MSS}.C_{fixe}$: pour faire circuler le jeton.

$N_{demande}(3C_{wireless}+C_{search}+2.C_{fixe})$: pour satisfaire une demande.

Dans cette application le coût est réduit aussi à de simples demandes faites par les mobiles, et est lié au nombre de demandes d'accès à la section critique.

IV.4.Problèmes propres à l'environnement mobile

Le principe des Deux-tiers est un bon outil pour l'adaptation des algorithmes classique à l'environnement mobile. Cependant, son utilisation ne génère pas des algorithmes parfaits, parcequ'il ne résous pas les problèmes liés à la mobilité elle même, tels que les déconnections, les reconnections, les Hand-off,...etc.

Dans la première application, si un mobile possédant l'autorisation d'accès se déplace vers une autre MSS que celle qui lui a délivré l'autorisation alors les conditions de service seront satisfaite dans cette nouvelle MSS aussi. Et par conséquent l'autorisation lui est envoyée une autre fois.

Dans la deuxième application, basée sur l'algorithme de Lann, le problème d'équité se pose dans le cas où un mobile se déplace devant le jeton, l'attend dans chaque céllule , accède à la région critique et se déplace vers la prochaine MSS dans l'anneau. Il pourra accéder, ainsi, $(N_{MSS}-1)$ fois à la région critique dans un seul tour du jeton. Et par conséquent un autre mobile peut attendre jusqu'à la satisfaction de $(N_{MSS}-1).(N_{mh}-1)$ demandes avant que le jeton ne lui soit délivré. Ce problème est lié à la mobilité des hôtes et n'est pas connu dans les anciens réseaux.

IV.5.Variables d'état ou communication de messages

Après avoir vu les inconvénients de l'application directe des algorithmes distribués classiques dans l'environnement mobile, et les avantages du principe des Deux-tiers, le choix entre le type d'algorithmes basés sur les variables d'état et celui des algorithmes basés sur la communication de messages est devenu évident.

L'utilisation des algorithmes utilisants les variables d'état dans une application directe est exclue, à cause du nombre excessif de messages qu'ils véhiculent entre les mobiles. Et par conséquent, le coût élevé au niveau de: l'utilisation du médium sans fil, la consommation d'énergie sur les mobiles, le nombre de messages véhiculés dans le réseau statique.

L'application du principe des Deux-tiers sur ce type d'algorithme ne fait que transférer la charge au réseau statique, où les algorithmes basés sur la communication de messages sont plus efficaces à cause du nombre réduit de messages qu'ils nécessitent.

V.CONCLUSION

Dans ce qui a précédé, on a vu les différents algorithmes d'exclusion mutuelle dans les systèmes distribués classiques et les différentes techniques qui y sont utilisées. On a introduit le principe des deux tiers et des exemples de son application. Enfin on a choisi , en ce qui concerne l'exclusion mutuelle classique, les algorithmes utilisant la communication de messages pour une éventuelle adaptation à l'environnement mobile.

Dans ce qui suit, on va choisir parmis ces algorithmes, le plus adéquat pour être appliqué dans un environnement mobile, en essayant de résoudre les problèmes, liés à la mobilité, qui vont apparaître

CHAPITRE III

CHOIX DES ALGORITHMES POUR L'ENVIRONNEMENT MOBILE

CHAPITRE 3: CHOIX DES ALGORITHMES POUR L'ENVIRONNEMENT MOBILE

I.INTRODUCTION

L'étude qu'on a faite de l'environnement mobile et du problème d'exclusion mutuelle permet de définir les critères nécéssaires pour choisir l'algorithme le plus adéquat pour résoudre un tel problème dans un tel environnement. Ces critères peuvent être perçus comme des buts à atteindre.

On cherche donc à:

-minimiser les opérations faites sur les mobiles pour conserver leur énergie,

-minimiser l'utilisation du médium sans fil vue sa limitation et son coût élevé,

-permettre aux mobiles de travailler en leurs différents modes,

-minimiser la charge sur le réseau fixe.

Tout cela en résolvant le problème d'exclusion mutuelle.

Dans un premier lieu on va faire des hypothèses sur le réseau fixe et sur le médium sans fil qui vont nous permettre de nous concentrer sur le problème principal qui est l'exclusion mutuelle. Ensuite, on choisira deux algorithmes, l'un pour l'affectation des canaux et l'autre pour l'accès aux sections critiques, parmi ceux vus dans les chapitres précédent en essayant de les améliorer.

II.MODELE ET HYPOTHESES

Le modèle, où vont se dérouler les algorithmes qu'on va choisir et appliquer, est le modèle décrit dans le chapitre précédent (IV.1). Vue que les algorithmes, objet d'étude, sont indépendants de l'architecture utilisée pour relier les entités fixes, ainsi que des protocoles utilisés pour assurer la communication entre les mobiles et les MSSs. On ne va pas s'intéresser aux problèmes de ces derniers tels que la perte des messages, l'altération...etc.

Pou cela, les hypothèses suivantes sont faites :

II.1.Réseau fixe

Le réseau fixe est supposé fiable c-à-d que les messages émis par un site fixe à un autre sont reçus correctement et dans l'ordre séquentiel de leur émission. On suppose, donc, que la transmission des messages via le réseau fixe se fait:

-sans perte.

-sans altération.

-sans deséquencement.

-sans duplication.

-sans régénération spontanée.

Ainsi, les primitives qu'on va utiliser, et qui traitent des problèmes cités précédement, dans les algorithmes sont l'envoie et la réception des messages transmis sans s'intéresser à leur mise en oeuvre.

Pour la résistance aux pannes on va s'intéresser à deux types de pannes:

-Pannes des MSSs.

-Pannes des mobiles.

Les pannes des liaisons de communication sont à la charge du réseau de transport. Le traitement des messages que se soit du médium filaire ou sans fil, dans un site, se fait séquentiellement, c-à-d que les messages sont reçus dans une file d'attente et traités l'un après l'autre. Les processus qui demandent l'accès à la section critique dans un site, accèdent aux différentes

variables en exclusion mutuelle (centralisée) avec le processus qui s'occupe du traitement des messages.

II.2.Médium sans fil

Vue que l'on s'intéresse aux effets de la mobilité des hôtes sur les algorithmes d'exclusion mutuelle, on ne va pas jusqu'aux détailles des primitives nécessaires pour envoyer et recevoir un message via le médium sans fil. Par contre on s'intéresse au problèmes liés à la mobilité: Hand-Off, déconnections, reconnection, ...etc.

III.Concurrence sur les canaux de communication

A cause de la mobilité des hôtes et de la limitation de la bande de fréquence réservée aux communications cellulaires dans un environnement mobile, la façon d'affecter les canaux de communications aux différentes cellules de manière à éviter les interférences, est très importante. Une solution qui garantie, d'une part, l'utilisation des canaux en exclusion mutuelle par les cellules adjacentes pour interdire toute interférence, et d'autre part, la diminution du temps de latence des mobiles dans le cas de leurs concentration dans des zones spécifiques, est souhaitée.

On a vu deux types de stratégies pour l'allocation des canaux de communication sans fil : statique et dynamique et le choix de la méthode dynamique est évident à cause de son efficacité (voir I.2.5.5.1). Ainsi que le choix de la méthode distribué par rapport a celle centralisée.

C'est dans la classe dynamique distribuée qu'on trouve l'algorithme de Singhal.

III.1.Algorithme

III.1.1.Principe

Initialement une MSS ne dispose d'aucun canal de communication. Pour en allouer, elle diffuse une demande à ses voisines immédiates pour déterminer les ensembles des canaux alloués, utilisés ou en état de transfert. Après avoir reçu ces informations, la MSS détermine s'il existe un canal libre

(qui n'est pas alloué par ses voisines). Si un tel canal existe, il sera alloué, sinon cela ne signifie pas qu'il n'en existe pas. En effet un canal alloué par une MSS voisine et qui n y'est pas utilisé peut être transféré. Pour éviter qu'une autre MSS voisine ne transfert ce canal en même temps, ce dernier n'est alloué que si ses voisines envoie un méssage d'accord.

III.1.2.Algorithme

Structure de données:

Chaque MSS M contient les informations suivantes:

Const

Spectrum: Ensemble des canaux disponibles dans le réseau ordonnés de la basse à la haute fréquence.

Var *Hor* : Entier initialisé à 0 (horloge de Lamport).

Allocate : Ensemble de canaux alloués à M. Initialisé à ∅.

Busy : Ensemble de canaux appartenant à *Allocate*, actuellement en fonctionnement (utilisés). Initialisé à ∅.

Transfer: Dans le cas où M ne dispose plus de canaux, et cherche à y allouer, elle cherche un canal appartenant à *Spectrum* mais n'est pas utilisé par ses voisines. Si un tel canal n'existe pas, elle cherche un canal alloué à l'une de ses voisines mais n'est pas utilisé (∈ *Allocate* - *Busy*).
Transfer Contient à chaque moment l'ensemble des canaux qui font objets de transfert de M vers ses voisines. Initialisé évidement à ∅.

Interfer1,Interfer2,Available,Free :Ensembles de canaux.

Defered : Ensemble de MSS.

Algorithme:

- Lorsqu'une MSS *M* veut allouer un canal à un mobile *Mh* :

  - *Available* := *Allocate* - *Busy* - *Transfer*;
  - Si *Available*≠∅ alors
    Debut
      - Choisir le canal K d'ordre élevé (haute fréquence) ∈ *Available*;
      - *Busy*:=*Busy* ∪ K; (1)
      - Allouer K à *Mh*

    Fin
  - Sinon
    Debut
      - Envoyer (REQUEST,*Hor*) à l'ensemble 'voisine' des voisines de *M* ;
      - *Interfer1*:=*Allocate*; *Interfer2*:=*Busy*∪*Transfer*;
      - *wait*:= card(voisine);

    Fin

- A la réception de (REPLY, $Allcate_{M'}$,$Busy_{M'}$,$Transfer_{M'}$) d'une MSS *M'*: voisine de *M* :
  - *wait--;*
  - *Interfer1*:=*Interfer*1∪$Allocate_{M'}$;
  - *Interfer2*:=*Interfer2*∪$Busy_{M'}$ ∪$Transfer_{M'}$;
  - Si *wait*=0 Alors
    Debut
      - *Free*:=*Srectrum-Interfer1*;
      - Si *Free* ≠ ∅ Alors
        Debut
          - Choisir le canal K d'ordre élevé (haute fréquence) ∈ *Free*;
          - *Allocate*:= *Allocate* ∪ K;
          - *Busy*:=*Busy* ∪ K; (2)
          - Allouer K à *Mh*

        Fin
        Sinon
        Debut
          - *Free*:= *Srectrum-Interfer2*;
          - Si *Free* = ∅ Alors Annuler la demande
            (il n'existe aucun canal libre)
            Sinon
            Debut
          - Choisir le canal K d'ordre bas (basse fréquence) ∈ *Free* ;
          - *Allocate*:= *Allocate* ∪ K;
          - *Busy*:=*Busy* ∪ K; (3)

- Envoyer (TRANSFER, K) à l'ensemble S des voisines ayants K∈*Allocate;*
- *wait:*=card(S); *accept*:=card(S);
- Fin

Fin

Fin

 A la réception de (AGREED,K): /*Resp.(REFUSE,K)

- *wait--; Accept--;* /*Resp.*Accept++*;
- Si *wait*=0 Alors

Debut

- Si *accept*=0 Alors

Debut

- Allouer K à *Mh*;
- Envoyer (RELEASE,K) à toutes les MSSs ∈ S;

Fin

Sinon

Debut

- *Allocate*:=*Allocate* - K;
- *Busy*:=*Busy* - K;
- *Free*:=*Free* - K;
- Envoyer (KEEP,K) à toutes les MSSs ∈ S;
- *Free:=Free*-K;
- Si *Free*≠∅ Alors

Debut

- Choisir le canal K d'ordre bas (basse fréquence) ∈ *Free* ;
- *Allocate*:= *Allocate* ∪ K;
- *Busy*:=*Busy* ∪ K; (4)
- Envoyer (TRANSFER, K) à l'ensemble S des voisines ayants K∈*Allocate;*
- *wait:*=card(S); *accept*:=card(S);

Fin

Sinon Annuler la demande
(il n'existe aucun canal libre à transférer)

Fin

Fin

- Envoyer (REPLY) à toutes les MSSs ∈ *Defered*

A la fin d'une session de communication utilisant un canal K

- *Busy* := *Busy* - K

A la réception de (REQUEST, $Hor_{M'}$) de *M'*

- Si un canal n'est pas demandé ou $Hor_{M'}$<*Hor* Alors

Envoyer (REPLY) à *M'*
Sinon *Defered* := *Defered* ∪ *M'*

□A la réception (TRANSFERT, K) de *M'*
- Si (K∈*Busy*) ou (K∈*Transfer*)
  Alors Envoyer (REFUSE, K) à *M'*
  Sinon
    Debut
      Transfer:= *Transfer* ∪ K;
      Envoyer (AGREED, K) à *M';*
    Fin

□A la réception de (RELEASE, K) de *M'*
- *Allocate*:=*Allocate* - K;
- *Transfer:=Transfer* - K;

□A la réception de (KEEP, K) de *M'*
- *Transfer*:= *Transfer* - K;

III.1.3.Propriétés

Ce qui est trés intérréssant dans cet algorithme est qu'il résoud le problème de l'allocation des canaux au niveau des MSSs (réseau fixe) sans faire appel ni aux mobiles ni au réseau sans fil. Le coût induit sur ces éléments est, donc, null.

Pour une station de base, l'algorithme ne nécessite que la contribution des stations voisines afin d'allouer un canal à un mobile donné. Ce qui permet de ne pas surcharger le réseau fixe comme dans le cas où on fait une gestion centralisé de l'ensemble des canaux.

Le nombre de messages nécessaires pour allouer un canal est:

0: si un canal existe déjà :*Allocate-Busy-Transfer*≠∅.

$2\times N_{voisines}$: s'il existe un canal non alloué par ses voisines.

$2\times N_{voisines}+(2\times N_{voisines}+Nvk)\times L$: S'il existes des canaux alloués par les cellules voisines et qui ne sont pas utilisé.

L: Nombre d'essai de transfert effectués pour allouer ce canal.

Nvk :Nombre de Stations voisines allouants le canal transféré.

Pour minimiser le coût de cette opération, on peut limiter le nombre de test de transfert par le nombre:

MaxTest=Min(Par,Card(*Free*));

Tel que « Par » est un paramètre prédéfinit qui détermine le nombre maximum d'essai de transfert pour ne pas augmenter le temps de latence des mobiles.

L'algorithme de Singhal ne nécessite pas beaucoup de calculs. Ses instructions ne contiennent que des opérations d'union, intersection, et de soustraction des ensemble des canaux. Ce qui peut être réduit à des opération logique OR, AND et XOR si les ensembles son représentés par des tableaux de bits égal à 1 si le canal appartient à l'ensemble et 0 sinon.

La charge de communication n'est supporté que par les stations de base voisines et non par le réseau fixe entier, ce qui permet l'ajout de nouvelles cellules sans augmenter le trafic dans le réseau.

III.1.3.1.Exclusion mutuelle

L 'algorithme garantie que les cellules voisines n'utilisent pas le même canal.

Preuve: Soit Nbr_i l'ensemble des cellules voisines à une cellule Ci. On doit prouver l'assertion suivante: $Busy_i \cap Busy_j = \varnothing, \ \forall \ Cj \in Nbr_i$.

Initialement, l'assertion est trivialement vérifiée car tous les ensemble sont vides. On a toujours $Busy_i \subseteq Allocate_i$. $Busy_i$ peut changer dans trois cas:

1. A l'étape (1) lorsque $Available_i \neq \varnothing$: soit k le canal choisi, $k \in Allocate_i$. En supposant que $Busy_i \cap Busy_j = \varnothing$ et $Allocate_i \cap Allocate_j = \varnothing$ avant l'allocation du canal k. On aura, aprés cette opération $\{Busy_i \cup k\} \cap Busy_j = \varnothing$. Ce qui vérifie l'assertion.

2. A l'étape (2) lorsque $Allocate_i = \varnothing$ et $Free_i \neq \varnothing$. Un canal $k \in Spectrum\text{-}(Allocate_i \cup_{j \in Nbri} Allocate_j)$ est ajouté à $Busy_i$ et $Allocate_i$. En supposant qu'il existe une autre cellule Cj qui utilise en même temps k. Cj ne transfere pas k à Ci tant que $k \in Busy_j$. Ceci implique que l'interférence supposée précedement ne peut arriver que si les ensembles *Allocate* des REPLY reçus par chacune de ces deux MSSs en provenance de l'autre ne contenait pas k. En se basant sur le schéma d'échange des méssages REQUEST et REPLY entre deux noeuds, les trois cas suivant peuvent arriver:

a. Ci a envoyé REPLY à Cj avant d'envoyer son REQUEST. Alors le REQUEST de Cj aura une estampille inférieure à celle de Ci, et lorsque le REQUEST de Ci arrive à Cj, elle diffère sa réponse destinée à Ci jusqu'à ce qu'elle prenne sa décision sur l'utilisation de k. Elle envoie ensuite $Allocate_j$

à Ci contenant k, et par conséquent k ne peut appartenir à $Free_j$ et ne sera pas choisi.

b. Cj a envoyé REPLY à Ci avant d'envoyer son REQUEST, ce qui est similaire au cas précédent. Ci choisi k et Cj ne le fait pas.

c. Chacune de Ci et Cj reçoit REQUEST après l'envoi de son REQUEST. Dans ce cas, la cellule ayant une estampille inférieure prend le canal et l'ajoute à son *Allocte* et l'envoi à l'autre cellule qui ne va pas choisir k.

Donc, deux cellules adjacentes ne peuvent pas allouer le même canal simultanément.

3. Aux étapes (3),(4), lorsque $Free_i \neq \varnothing$. Dans ce cas le canal n'est alloué que si toutes les cellules voisine répondent par AGREED sinon le canal est rejeté et un autre est choisi. Donc l'assertion est vérifiée et l'algorithme garantie l'exclusion mutuelle.

L'utilisation de la technique d'estampille de Lamport permet d'ordonner globalement les REQUEST ce qui permet d'éviter tout circuit et par conséquent tout interblocage.

III.1.4.Résistance aux pannes

L'algorithme résiste bien aux pannes des MSSs, les panne des mobiles ne l'affectant pas. Le traitement de la panne d'une MSS ne nécéssite que l'envoie d'un méssage vers ses voisines pour les informer de sa pannes. Ces dernières vont l'éliminer de leur ensemble de voisines. Les canaux qui étaient utilisés par cette MSS seront récupérés automatiquement lorsque ces voisines les trouvent dans *Spectrum- Allocate*$_j$.

IV.EXCLUSION MUTUELLE CLASSIQUE

On a vu dans le chapitre précédent le principe des « deux tiers » qui permet de transférer la charge d'un algorithme distribué des mobiles aux MSSs, qui sont plus puissantes et dont le coût de communication est relativement faible.

Avant l'application de ce principe, les critères du choix d'un algorithme distribué pour son utilisation dans l'environnement mobile étaient la consommation d'énergie sur les mobiles, la surcharge du médium sans fil par les messages nécéssaires à cet algorithme, les déconnections des mobiles,... etc.

Après l'application de ce principe, les mobiles ne font qu'envoyer des demandes aux stations de base qui, entre elles, décident de donner l'autorisation d'accès à la section critique à ce mobile ou à un autre. Et par conséquent, les critères du choix ne concernent plus que la partie fixe du réseau: le nombre de messages échangés, la résistance aux pannes,... etc.

On a deux types d'algorithmes d'exclusion mutuelle: ceux utilisant les variables d'état, et ceux utilisant la communication de messages. On a vu dans le chapitre précédent que les algorithmes à variables d'état nécessitent un nombre élevé de messages pour assurer l'accès en exclusion mutuelle à la section critique. Et que les algorithmes basés sur la communication de messages sont plus adaptés à l'environnement mobile. Ces derniers se divisent en deux classes selon la technique utilisée:

-Distribution d'une file d'attente: Cette classe englobe les algorithmes qui distribuent une file, FIFO, contenant les informations relatives aux différentes demandes dans toutes les MSSs, pour permettre une vue globale à chaque site (Lamport 78, Ricart & Agrawala). Pour garantir la cohérence de cette file, ces algorithmes utilisent un nombre important de messages. Par contre, ils ne nécessitent pas de localisation pour envoyer l'autorisation d'accès aux mobiles, car toutes les demandes se trouvent au niveau de chaque station de base. Ainsi ces algorithmes résistent bien aux pannes des MSSs: Au cas où une MSS tombe en panne, les mobiles locaux peuvent se déplacer vers d'autre station où ils reçoivent les autorisations.

-Circulation d'un jeton: Cette classe englobe les algorithmes utilisants la technique du jeton (Lann, Susiki & Kasami). Ces algorithmes nécessitent un nombre de message relativement inférieur, mais, ils connaissent un certain nombre de problèmes tels que la perte du jeton et sa régénération. Contrairement à ceux de la classe précédente, ces algorithmes nécessitent la localisation des mobiles pour leur envoyer l'autorisation d'accès, et si une station de base tombe en panne les demandes qui y sont déposées seront perdues.

Si on prend en compte que:

-Le coût de localisation d'un mobile est au maximum $N_{MSS}.C_{fixe}$: Coût de diffusion de la requête de recherche et de réception de la réponse ,et que les techniques de localisation *Inform* et *Proxy* vues dans le premier chapitre peuvent le diminuer.

-Dans les réseaux mobiles, les stations de bases sont souvent dupliquées dans les mêmes cellules pour tolérer leur pannes. Et par conséquent, les pannes des MSSs sont rares et considérées comme des cas particuliers.

Et que, comme on va le voir dans ce qui suit, les inconvénients de cette deuxième classe deviennent moins importants voire négligeables. Alors, le choix des algorithmes de la deuxième classe c-à-d utilisants la technique du jeton devient évident.

Cette classe contient deux algorithmes : celui de Lann et celui de Susiki & Kasami. Le premier fait circuler le jeton, qui est une information particulière, fixe et connue par toutes les MSSs à tout instant, entre toutes les MSSs dans un anneau fixe qui n'est reconfiguré qu'en cas de panne d'une station de base. Cette solution pose le problème de nombre infini de messages: Au cas où aucun mobile ne demande d'accéder à la section critique, le jeton continue à circuler dans l'anneau. Cela représente une surcharge inutile pour le réseau fixe.

En plus, si les demandes de la section critique sont concentrées dans un ensemble de MSSs, alors que les autres ne contiennent pas de demandes, l'envoi du jeton à ces dernières ne fait qu'augmenter le temps de latence des mobiles demandant l'accès à la section critique.

Ces problèmes sont résolus, dans le deuxième algorithme (Susiki & Kasami), en accompagnant le jeton d'informations concernant l'évolution des demandes au niveau des différentes stations de base. Cela permet d'éviter l'envoie du jeton aux MSSs qui ne le demandent pas.

Pour toutes ces raisons, on a choisi l'algorithme de Susiki & Kasami pour la résolution du problème de l'accés en exclusion mutuelle à la section critique en environnement mobile. Aprés les modifications nécéssaires on obtient l'algorithme suivant:

IV.1.Algorithme

IV.1.1.Principe

Le jeton, circulant entre les MSSs dans le sens croissant de leur indices, contient les numéros des dernières requêtes satisfaites sur chaque MSSs. Chaque mobile désirant entrer en section critique envoie une demande et une seule à la MSS qui le gère en ce moment. Chaque MSS a, donc, une file FIFO des demandes d'accés en section critique. A chaque fois qu'une MSS reçoit sa première demande, aprés le dernier passage du jeton, elle diffuse une requête accompagnée de son horloge (incrémenté juste avant son envoie) à toutes les autres MSS (ne sachant pas où se trouve le jeton) pour réclamer le jeton.Ainsi, cette requête est enregistrée au niveau de ce dernier. A la réception du jeton, une MSS envoie les autorisations d'accès aux mobiles qui y ont déposé des demandes à tour de rôle dans l'ordre FIFO de leur arrivée, bien sûr après leur localisation.

Après la satisfaction des demandes locales, une MSS M_i met à jour son champ dans le jeton pour annuler sa requête. Elle envoie ensuite le jeton à la première de ses voisines M_j dans l'ordre $M_{i+1},..,M_N,1..M_{i-1}$ ayant jeton[j]>requête[j] c-à-d ayant réclamé le jeton. L'ordre i+1,...,N,1,...,i-1 est très important pour éviter la famine des MSSs en cas où leur prédécesseurs demandent toujours le jeton. Les demandes qui arrivent, aux niveau d'une MSS, au moment où elle possède le jeton seront mises dans une autre file 'des demandes différées' et ne seront satisfaites qu'au prochain tour du jeton.

IV.1.2.Algorithme

♦ Actions exécutées par un mobile MH:

- Pour accéder à la section critique
  - Envoyer (DEMANDE_SC, MH) à la MSS courante M
- A la réception de (AUTORISATION) de M
  - <Section Critique>
  - Envoyer (LIBERER, MH) à M

♦ Actions exécutées par une MSS M:

Var
Hor: 0..+∞ init 0;
JetonPresent,Dedans: boolean init False;
Requete,Jeton: Tableau[1..N_{MSS}] de 0..+∞ init 0;
FileDemandes,FileService: File FIFO de MH init ∅;

- A la réception de (DEMANDE_SC,MH) d'un mobile MH.
  - Enfiler(FileDemandes,MH);
  - Si ¬JetonDemande Alors
    Debut
    JetonDemande:=True;
    - Si ¬JetonPresent Alors
      Debut
      - Hor:=Hor+1;
      - Diffuser(REQ,Hor,M) à toutes les autres MSSs;

      Fin
      Sinon
      Debut
      - Dedans:=True;
      - JetonPrésent:=True;
      - FileService:=FileDemandes;
      - FileDemandes:=∅;
      - Défiler(FileService,MH);
      - Chercher(MH);
      - Envoyer(AUTORISATION,M) à MH;

      Fin

- A la réception du jeton
  - Dedans:=True;
  - JetonPrésent:=True;
  - FileService:=FileDemandes;
  - FileDemandes:=∅;
  - Défiler(FileService,MH);
  - Chercher(MH);
  - Envoyer(AUTORISATION,M) à MH;

- A la réception de (REQ,Hor',M') de M'
  - Requete[M']:=Max(Hor',Requete[M']);
  - Si (JetonPresent) & (¬ Dedans) Alors
    Debut
      - Pour j de M+1..N_{MSS},1..M-1 Faire
        Debut
          - Si (Requete[j]>Jeton[j])&(JetonPresent) Alors
            Debut
              - JetonPresent:=False;
              - Envoyer(Jeton) à j;
              - Exit;

            Fin

        Fin

    Fin

- A la réception de (LIBERER, MH) de MH
  - Si ¬(filevide(FileService)) alors
    Debut
      - Défiler(FileService,MH');
      - Chercher(MH');
      - Envoyer(AUTORISATION,M) à MH';

    Fin
    Sinon
    Debut
      - Dedans:=False;
      - Jeton[M]:=Hor;
      - j:=M+1;
      - Pour j de M+1..N_{MSS},1..M-1 Faire
        Debut
          - Si (Requete[j]>Jeton[j])&(JetonPresent) Alors
            Debut
              - JetonPresent:=False;
              - Envoyer(Jeton) à j;
              - Exit;

            Fin

        Fin

```
•Si ¬(filevide(FileDemandes)) Alors
  Debut
      •Si JetonPresent Alors
        Debut
            •Dedans:=True;
            •FileService:=FileDemandes;
            •FileDemandes:=∅;
            •Défiler(FileService,MH);
            •Chercher(MH);
            •Envoyer(AUTORISATION,M) à MH;
        Fin
       Sinon
        Debut
            •Hor++;
            •Diffuser(REQ,Hor,M) à toutes les MSS;
        Fin
  Fin
Fin
```

IV.1.3.Propriétés

L'exclusion mutuelle est trivialement garantie par l'algorithme, car le jeton ne peut se trouver que dans une seule MSS à la fois, et cette MSS ne peut envoyer qu'une seule autorisation à la fois, durant sa possession du jeton. Et par conséquent, à tout instant donné, un seul mobile au plus peut avoir une autorisation d'accès à la section critique.

L'utilisation de l'algorithme proposé ne peut conduire à des cas de famine, c-à-d qu'il ne peut y avoir des demandes qui attendent l'autorisation d'accès indéfiniment. Puisqu'on ne sert que les demandes envoyées avant l'arrivée du jeton, ce dernier ne peut rester indéfiniment dans une céllule et puisque le jeton est envoyé à la première MSS suivante dans l'anneau l'ayant demandé. Alors toute MSS ayant réclamé le jeton, le recevra en un temps fini.

L'ajout de nouvelles stations de base ou de sites fixes se fait facilement en augmentant la taille des vecteurs Jeton et Requête de chaque station. Et aucun changement de l'algorithme n'est nécessaire

Pour éviter la croissance possibles des variables Hor, et par conséquent, les champs des vecteurs Requête et Jeton. Si une MSS M, qui reçoit le jeton, remarque que son Hor et devenue assez grande, elle exécute:

Jeton[M]:=0;

Diffuser(INITREQ,M);

Une MSS qui reçoit ce message exécute:

Requête[M]:=0;

IV.1.3.1.Coût de communication

Le coût induit par cet algorithme pour satisfaire une demande est:

$$C_{wireless}+2(C_{wireless}+C_{fixe})+(N_{MSS}-1)C_{fixe}+C_{fixe}+C_{search}$$

- $C_{wireless}$: Coût d'envoi de la demande du mobile à la MSS
- $2(C_{wireless}+C_{fixe})$: Coût d'envoie des messages d'autorisation et de libération
- $(N_{MSS}-1)C_{fixe}$: Coût[2] de diffusion de la demande
- C_{fixe} : Coût d'envoie du jeton d'une MSS à une autre
- C_{search} : Coût de localisation du mobile

Le coût de satisfaction d'une demande est donc:

$$3C_{wireless}+(N_{MSS}+2)C_{fixe}+C_{search}$$

Le coût pour k demandes sera, dans le pire des cas où chaque MSS contient une seule demande au moment de l'arrivé du jeton:

$$k(3C_{wireless}+(N_{MSS}+2)C_{fixe}+C_{search})$$

Et le coût de satisfaction de k demandes dans la même MSS est:

$$3kC_{wireless}+(2k+N_{MSS})C_{fixe}+kC_{search}$$

Pour les mobiles, l'application du principe des deux tiers a permit de réduire leur consommation d'énergie. Chaque mobile ne consomme de l'énergie que pour envoyer deux messages (sans fil) et en recevoir un. En plus un mobile qui dépose une demande, dans une MSS, peut se mettre en mode veille pour conserver son énergie. Il ne sera réveillé que lorsqu'il recevra l'autorisation d'accés en section critique.

Les mobiles qui ne déposent pas de demandes ne sont pas concernés par l'exécution de l'algorithme.

[2] *Si c'est la première demande aprés la dernière visite du jeton à cette MSS sinon le coût est égal à 0.*

IV.2.Comparaison avec les autres algorithmes

Dans l'algorithme de Lamport adapté à l'environnement mobile et basé sur une file distribué pour éviter la localisation, la gestion et la préservation de la cohérence de la file exige un nombre élevé de messages.

Ainsi, le coût de satisfaction d'une demande était de:

$$3C_{wireless} + 4(NMSS-1)Cfixe$$

Dans notre algorithme au pire des cas ce coût est:

$$3C_{wireless} + (2N_{MSS} + 2)C_{fixe}$$

c-à-d qu'on gagne, au minimum, $(2N_{MSS} - 6)$ messages dans le réseau fixe. Ce qui veut dire que dés que le réseau contient plus de 3 MSS, notre algorithme devient moins coûteux.

Dans l'algorithme de Lann adapté à l'environnement mobile, le coût de satisfaction d'une demande était de:

$$3C_{wireless} + (N_{MSS} + 2)C_{fixe}$$

Mais si aucun mobile ne demande la section critique, le nombre de messages peut être infini. Pour cela, on préfère perdre un coût égale à $(N_{MSS} + 1)C_{fixe}$ dans notre algorithme pour éviter un tel cas.

Il faudrait noter que, dans notre algorithme, plus les mobiles se concentrent dans les mêmes céllules, plus le coût induit sur le réseau fixe tends vers C_{fixe}.

IV.2.Perte de jeton :

Comme on l'avait déjà mentionné, ce type d'algorithme pose le problème de perte de jeton due, dans notre cas, à la panne d'une MSS au moment où elle détient le jeton ou à la panne d'un mobile ayant l'autorisation d'accès.

IV.2.1.Panne d'une MSS

Lorsqu'une MSS M tombe en panne, le réseau de transport délivre le message (PANNE,M) aux autres MSSs pour leur signaler cette panne. A la réception de ce message, il faut immédiatement déterminer si M avait le jeton lors de sa panne ou non. Si oui alors il est régénéré par celle qui le lui a délivré, pour garder les informations circulant avec le jeton. Pour définir cette MSS un autre champ Version Jeton est ajouté au jeton. Ce champ est

incrémenté par chaque MSS avant la transmission du jeton à son successeur. La MSS ayant la dernière version du jeton le régénère.

Pour cela, la MSS qui précède M dans l'anneau et qui n'est pas en panne initialise un message (ELECTION) et le fait circuler dans l'anneau pour, en même temps, définir si M avait le jeton au moment de sa panne et définir celle qui doit le régénérer au cas échéant.

Il est à noter que chaque MSS contient un tableau de booléen Panne tel que la case Panne[i] prend la valeur Vrai si la MSS i est en panne.

IV.2.1.1.Protocole

♦Actions exécutées par une MSS M:

 A la réception de (PANNE,M')
- Panne[M']:=True;
- Si M=PPNEP(M') Alors
  Debut
  - Si ¬JetonPresent Alors
    Envoyer(ELECTION,VersionJeton,M,M) au PSNEP(M);

  Fin

 A la réception de (REGENERER)
- JetonPresent:=true;
- Pour j de M+1,..N_{MSS},1..M-1 Faire
  Debut
  - Si (Requete[j]>Jeton[j])&(JetonPresent) Alors
    Debut
    - JetonPresent:=False;
    - Envoyer(Jeton) à j;
    - Exit;

    Fin

  Fin

```
 A la réception de (ELECTION,VersionJeton',M',M'')
    •Si M=M'' Alors  /* Le jeton est perdu */
      Debut
        •Si M=M' Alors  /* M est régénératrice */
          Debut
            •JetonPresent:=true;
            •Pour j de M+1,..N_MSS,1..M-1 Faire
              Debut
                •Si(Requete[j]>Jeton[j])&(JetonPresent)Alors
                  Debut
                    •JetonPresent:=False;
                    •Envoyer(Jeton) à j;
                    •Exit;
                  Fin
              Fin
          Fin
         Sinon Envoyer(REGENERER) à M';
    Sinon
      Debut
        •Si ¬JetonPresent Alors
          Debut
            •Si VersionJeton>VersionJeton' Alors
             Envoyer(ELECTION,VersionJeton,M,M'')
            au PSNEP(M);
             Sinon
            Envoyer(ELECTION,VersionJeton',M',M'')
            au PSNEP(M);
          Fin
      Fin
```

La fonction PSNEP(M) premier succésseur non en panne (resp. PPNEP(M) premier prédécesseur non en panne) fournit l'identité de la première MSS successeur (resp. prédécesseur) de M qui n'est pas en panne.

```
Fonction PSNEP(M:MSS);
      Debut
      j:=M+1;
      fin:=false;
      Tant que ¬Fin Faire
           Debut
           Si ¬Panne[j] Alors
              Debut
               fin:=true;
               PSNEP:=j;
               Fin
           j:=(j+1)mod N_MSS;
           Fin
      Fin

Fonction PPNEP(M:MSS);
      Debut
      j:=M-1;
      fin:=false;
      Tant que ¬Fin Faire
           Debut
           Si ¬Panne[j] Alors
              Debut
               fin:=true;
               PPNEP:=j;
               Fin
           j:=(j-1)mod N_MSS;
           Fin
      Fin
```

Lors de la panne d'une MSS, les demandes des mobiles qui y sont déposées seront perdues. Par conséquent ces mobiles seront bloqués car il attendent une réponse de la MSS en panne. Pour régler ce problème un mobile qui dépose une demande dans une MSS sauvegarde son identité et l'envoie à chaque hand-off avec le message de demande d'accueil pour informer la MSS courante de cette demande. Lorsque cette MSS reçoit le message (PANNE,M') du réseau de transport, elle cherche dans sa liste des mobiles locaux ceux ayant des demandes chez M'. Si de tels mobiles existent, elle leur envoie le message (PANNE, M'). A la réception de ce message, un mobile peut se trouver dans trois cas:

-Il attend l'autorisation: alors il doit faire une nouvelle demande.

-Il l'a déjà reçu: alors il doit abandonner la section critique, ses modification sont annulées et il doit faire une nouvelle demande.

-Il a libéré la section critique et a envoyé le message de libération: alors il ne fait rien.

IV.2.2.Panne d'un mobile

A tout instant, un mobile connecté au réseau peut tomber en panne ou être inaccessible (quitter la surface couverte par le réseau). Si un mobile tombe dans un cas pareil au moment où il a l'autorisation d'accès à la section critique alors l'algorithme sera bloqué car la MSS ayant le jeton attendra indéfiniment la réponse de ce mobile. Et par conséquent aucun autre mobile ne peut accéder à la séction critique. Pour pallier à ce problème, une MSS qui envoie l'autorisation à un mobile initialise un temps maximum d'attente de la réponse. Si, après l'écoulement de ce temps, le mobile n'a pas libéré la ressource alors l'autorisation lui est retirée et un message lui est transmis pour l'informer de ce retrait: immédiatement s'il est connecté sinon après sa reconnection.

Un mobile ayant reçu l'autorisation d'accés à la séction critique ne peut se déconnecter qu'après sa libération.

IV.3.Accès juste au jeton

Selon leurs déplacements, des mobiles peuvent accéder à la section critique plusieurs fois alors que d'autres qui l'ont demandé avant eux n'y ont pas accédé.

Supposons deux mobiles Mh1 et Mh2 qui se trouvent respectivement sous le contrôle des MSSs M1,M3 et envoient simultanément des demandes d'accès à la ressource, chacun à sa MSS. Si Mh1 reçoit l'autorisation d'accès dans la cellule gérée par M2, et envoie une autre demande à M2 avant que le jeton n'y arrive, alors il reaccédera à la ressource alors que Mh2 n'y a pas encore accédé.

Pour palier à ce problème, une MSS qui envoie une autorisation à un mobile accompagne ce message par l'identité d'une autre MSS. Le mobile devra attendre que cette dernière reçoive le jeton au moins une fois pour faire une autre demande.

Un mobile qui envoie une demande à une MSS doit envoyer avec cette demande le uplet (M, NORequete). Lorsque cette MSS décide de lui envoyer le jeton elle vérifie avant, si la requête numéro NoRequête de M a été satisfaite. Si c'est le cas, elle lui envoie l'autorisation sinon elle retarde cet envoie à la prochaine visite du jeton.

IV.4.Localisation

Comme il a été déjà indiqué, l'utilisation de la technique du jeton nécessite la localisation des différents mobiles pour leur envoyer les autorisations. En effet un mobile ayant déposé une demande d'accès chez une MSS peut se déplacer vers une autre où il doit recevoir son autorisation d'accès.

Dans notre algorithme, la localisation est réalisée par la fonction Chercher (Mh: Mobile) qui doit retourner l'identité de la dernière MSS fréquentée par Mh et l'état actuel de ce mobile: connecté, déconnecté, panne ou veille.

Pour réaliser cette fonction, on a recourt aux méthodes présentées dans le premier chapitre (*Search*, *Inform*, *Proxy*). Le choix entre ces méthodes nécessite une étude du comportement général des mobiles.

IV.4.1.Méthode *Search*

Cette stratégie consiste à chercher un mobile donné dans toutes les MSSs à chaque fois qu'il est suicité. Le coût induit par cette méthode est indépendant des mouvements des mobiles après le dépôt de leur demandes.

```
Fonction Rechercher(MhId: Mobile)
      Debut
            •Si MhId ∈ LocalList Alors
              Debut
                    •M':=M;
                    •Etat:=Etat(MhId)
              Fin
             Sinon
              Debut
                    •Diffuser (CHERCHER,M,MhId);
                    •Attendre la réception de (DECLARE,M',MhId,Etat) de
                     M';
              Fin
            •Retourner (M',Etat)
      Fin.

□A la réception de (CHERCHER,M', MhId') de M'
      Si MhId' ∈ LocalList alors
       Envoyer (DECLARE,M,MhId',Etat(Mh')) à M'
```

IV.4.2.Méthode *Inform*

Dans la méthode *Inform*, chaque MSS connaît à tout instant la localité des mobiles ayants déposé des demandes chez elle . Après le dépôt d'une demande dans une MSS donnée un mobile doit informer cette MSS de sa localité à chaque changement de cellule (Hand-Off).

```
Fonction Rechercher (MhId:Mobile)
      Debut
            •Retourner(Localité(MhId),Etat(MhId))
      Fin.
*Lors des Hand-Off
      □A la réception de (DEMANDEACCUEIL,MhId,M',M'')
       Si M''≠ NIL Alors
            Envoyer(IlEstLa,MhId,M,Connecté) à M''

      □A la réception de (IlEstLa,MhId,M',Etat) de M'
        Localité[Mh]:=M';
        Etat[MhId]:=Etat;
```

♦ Actions Exécutées par une MSS *Proxy* M

€ A la réception de (CHERCHER,M',MhId) de M';
- Si Proxy[MhId]=M Alors
  Debut
  - Si MhId∈LocalList Alors
    Envoyer(DECLARE,M,MhId,Etat(MhId));
    Sinon Envoyer (CHERCHER,M',MhId) à toutes les MSS locales;

  Fin
  Sinon
  Envoyer (LOCALISE,M',MhId) à Proxy[MhId];

€ A la réception de (LOCALISE,M',MhId)
- Envoyer(CHERCHER,M',MhId) à toutes les MSSs locales;

♦ Actions exécutées par toute MSS M

€ A la réception de CHERCHER(M',MhId) du *Proxy*
- Si MhId∈LocalList Alors
  Envoyer (DECLARE,M,MhId,Etat(MhId)) à M';

* Lors des Hand-Offs

€ A la réception de DEMANDEACCUEIL(MhId,M',M'') de MhId
/*M': la MSS où MhId se trouve actuellement.
/*M'': la MSS où MhId a déposé sa demande de séction critique.
- Si Proxy(M)≠Proxy(M') Alors
  Envoyer (CHANGEPROXY,MhId,Proxy(M)) à Proxy(M'');

€ A la réception de (CHANGEPROXY,MhId,M') de M'
- Proxy(MhId):=M';

Pour choisir entre la méthode *Search* et la méthode *Inform*, supposons un mobile Mh qui dépose une demande dans une MSS M et fait N_{move} déplacements entre les cellules. En utilisant la stratégie *Inform* Mh va informer M N_{move} fois, c-à-d que chaque MSS qui reçoit Mh envoie un message à M pour l'informer de la nouvelle localité de Mh. Le coût de localisation sera $N_{move}.C_{fixe}$.

Inform est meilleure que *Search* si $N_{move}.C_{fixe} < N_{MSS}.C_{fixe}$

$\Rightarrow N_{move} < N_{mss}$

Si le nombre de déplacements intercellulaires de chaque mobile est inférieur au nombre total des MSSs, alors il sera préférable d'utiliser la méthode *inform*, et vis versa.

IV.4.3.Méthode combinée (*Proxy)*

Si les mobiles font beaucoups de mouvements intercellulaires, mais le font entre des cellules spécifiques. Alors, on regroupe ces dernières en sous-ensembles géré chacun par une MSS principale (*Proxy*). Un mobile n'informe son *Proxy* (où il a déposé une demande) que dans le cas des déplacements inter-*Proxys*, et sera cherché dans les cellules gérées par ce *Proxy*.

Lorsqu'un mobile dépose une demande d'accès en sèction critique, il le fait dans une MSS M qui en informe sa proxy qui gère une variable indiquant la proxy où se dernier évolue.

La fonction « rechercher » est exécutée par la MSS où le mobile a déposé sa demande de section critique au moment où elle décide de lui envoyer une autorisation d'accès.

```
♦Fonction Rechercher(MhId: Mobile)
    Debut
    •Si MhId ∈ LocalList Alors
      Debut
          •M':=M;
          •Etat:=Etat(MhId)
      Fin
     Sinon
      Debut
          •Envoyer (CHERCHER,M,MhId) à la MSS Proxy(M);
          •Attendre la réception de (DECLARE,M',MhId,Etat) de M';
      Fin
    •Retourner (M',Etat)
    Fin.
```

Le coût induit par cette méthode pour un mobile faisant $N_{proxymove}$ déplacements inter-*Proxys* est :

$$N_{proxymove}.C_{fixe} + N_{MSSproxy}.C_{fixe}$$

Informations des déplacements inter-Proxys

Recherche dans le même *Proxy*

$N_{MSSproxy}$: Nombre de MSS gérées par un proxy (N_{MSS}/N_{Proxy}).
$N_{Proxymove}$:Nombre de déplacements inter-proxy.

Le choix entre la méthode *Proxy* et la méthode *Inform* se fait comme suit:

Inform est meilleure que *Proxy* si

$N_{move}.C_{fixe} < N_{proxymove}.C_{fixe} + N_{MSSproxy}.C_{fixe}$
$\Rightarrow N_{move} < N_{proxymove} + N_{MSSproxy}$
$\Rightarrow N_{move} < N_{proxymove} + N_{MSS}/N_{proxy}$

Lorsque le nombre de mouvements inter-cellulaires fait par un mobile est inférieur au nombre des mouvements inter-*Proxys* ajouté au nombre des MSS dans ce *Proxy* alors il serait préférable d'appliquer la méthode *Inform* et vis versa.

La méthode *search* est meilleure que *Proxy* si

$N_{MSS} < N_{proxymove} + N_{MSS}/N_{proxy}$

$$\Rightarrow N_{MSS} < \frac{N_{Proxymove}}{1-\left(1/N_{Proxy}\right)}$$

Donc, en faisant des observations sur les mouvements des mobiles dans un temps suffisant pour déterminer leur comportement général, on peut choisir la stratégie qui s'y adapte le mieux.

IV.5.Sites fixes

Le réseau fixe est un réseau distribué normal avec un ensemble de MSSs. Cela veut dire qu'il contient des sites fixes qui utilisent déjà les services du réseau. Lors de la conception des algorithmes de l'environnement mobile, ils faut prendre en compte ces sites parce qu'ils peuvent utiliser les mêmes ressources que les hôtes mobiles.

Dans l'algorithme proposé, les sites fixes sont considérés comme des MSS qui ne seront jamais visitées par les mobiles. Ils demandent le jeton pour satisfaire une seule demande locale.

Le protocole d'accès à la section critique par un site fixe sera comme suit:

♦ Actions exécutées par un site fixe

Var
Hor: 0..+∞ init 0;
JetonPresent,Dedans: booléen init False;
Requete,Jeton: Tableau[1..$N_{MSS+sitesfixes}$] de 0..+∞ init 0;

□Pour accéder à la ressource
- Si ¬JetonPresent alors
  Debut
  - Hor:=Hor+1;
  - Diffuser(REQ,Hor,M) sur tout le réseau fixe;
  - Attendre la réception du jeton;

  Fin
- Dedans:=True;
- JetonPrésent:=True;

<Section Critique>

- Dedans:=false;
- Pour j de M+1,..$N_{MSS+sitesfixes}$,1..M-1 Faire
  Debut
  - Si (Requete[j]>Jeton[j])&(JetonPresent) Alors
    Debut
    - JetonPresent:=False;
    - Envoyer(Jeton) à j;
    - Exit;

    Fin

  Fin

□A la réception de (REQ,Hor',M') de M'
- Requete[M']:=Max(Hor',Requete[M'];
- Si (Jeton Present) & (¬ Dedans) Alors
  Debut
  - Pour j de M+1,..$N_{MSS+sitesfixes}$,1..M-1 Faire
    Debut
    - Si (Requete[j]>Jeton[j])&(JetonPresent) Alors
      Debut
      - JetonPresent:=False;
      - Envoyer(Jeton) à j
      - Exit;

      Fin

    Fin

  Fin

IV.6.Algorithme final

♦ Actions exécutées par un mobile MH

□VAR

$M_{depôt}$:MSS init NIL;
/* La MSS où MH à déposé une demande */
$M_{attente}$:MSS init NIL;
/* La MSS que MH doit attendre la satisfaction de sa requêtte numéro Nreq */
Nreq :0..+∞ init 0;
SecDemandé,AutReçue :Boolean;

□Pour accéder à la section critique

- Si ¬SecDemandée Alors
  Debut
  SecDemandée:=True;
  $M_{Depôt}$:=M
  - Envoyer (DEMANDE_SC, MH,$M_{Attente}$,Nreq) à la MSS courante M

  Fin

□A la réception de (AUTORISATION,M',Nr) de M

- $M_{Attente}$:=M';
- Nreq:=Nr;
- AutReçue:=True;

<Section Critique>

- Envoyer (LIBERER, MH) à M
- SecDemandée:=False;
- AutReçue:=False;

□A la réception de (PANNE,M') de M

- Si ¬AutReçue Alors Libérer la section critique;
- AutRecue:=False;
- $M_{Depôt}$:=M;
- Envoyer(DEMANDE_SC, MH,$M_{Attente}$,Nreq) à la MSS courante M

□Lors de Hand-Off

- Envoyer(DEMNDEACCUEIL,MH,Mlocale,$M_{Depôt}$) à la MSS courante

□Pour se déconnecter

- Si ¬AutReçue Alors
  Envoyer(DISCONNECT,MH) à Mlocale;

♦ Actions exécutées par une MSS M

```
Var
    Hor: 0..+∞ init 0;
    JetonPresent,JetonDemande,Dedans: booléen init False;
    Requete,Jeton: Tableau[1..N_MSS+N_fixe] de 0..+∞ init 0;
    FileDemandes,FileService: File FIFO de MH init ∅;
    Panne : Tableau [1..N_MSS+N_fixe] de booléen init false;

□A la réception de (DEMANDE_SC,MH) d'un mobile MH.
    Enfiler(FileDemandes,MH);
    Si ¬JetonDemande Alors
      Debut
        JetonDemande:=True;
        Si ¬JetonPresent alors
          Debut
            Hor:=Hor+1;
            Diffuser(REQ,Hor,M)
          Fin
        Sinon Aller à la réception du jeton.
      Fin

□A la réception de (REQ,Hor',M') de M'
    Requete[M']:=Max(Hor',Requete[M'];
    Si (Jeton Present) & (¬ Dedans) Alors
      Debut
        Pour j de M+1,..N_MSS,1..M-1 Faire
          Debut
            Si (Requete[j]>Jeton[j])&(JetonPresent) Alors
              Debut
                JetonPresent:=False;
                 Envoyer(Jeton) à j
              Fin
          Fin
      Fin
□A la réception du jeton
    Var F: File init ∅;
    Dedans:=True;
    JetonPrésent:=True;
    Tantque ¬Filevide(FileDemandes) faire
      Debut
        Défiler(FileDemandes,MH,Matt,Nr);
        Si (Matt≠Nil)∧(Jeton[Matt]<Nr) Alors
                  Enfiler(F,MH,Matt,Nr);
        Sinon Enfiler(FileService,MH);
       Fin
    FileDemandes:=F;
```

```
        Sinon
          Debut
            Dedans:=False;
            Jeton[M]:=Hor;
            Pour j de M+1,..N_MSS+N_fixe,1..M-1 Faire
              Debut
                Si (Requete[j]>Jeton[j])&(JetonPresent) Alors
                  Debut
                    JetonPresent:=False;
                     Envoyer(Jeton) à j
                  Fin
              Fin
            Si ¬(filevide(FileDemandes)) Alors
               Debut
                 Si JetonPresent Alors Aller à réception de jeton
                 Sinon
                   Debut
                       Hor:=Hor+1;
                       Diffuser(REQ,Hor,M)
                   Fin
               Fin
             Sinon JetonDemande:=False;
          Fin

□A la réception de (PANNE,M')
        Protocole de détection de perte et de régénération du jeton
        Pour tout MH ∈LocalList ∧ MH.Demande =M'
              Envoyer(PANNE,M') à MH;
```

Les sites fixe exécutent les mêmes actions déjà présentées en plus ils doivent contribuer à la détection de la perte et la régénération du jeton.

V.CONCLUSION

Après le choix des algorithmes et le traitement des différents problèmes qui sont apparus, il ne nous reste plus qu'à implémenter l'algorithme final auquel on a aboutie. Puisqu'un réseau mobile n'existe pas encore, la simulation est indispensable pour démontrer pratiquement la validité de ces algorithmes. La mise en oeuvre se fait par la simulation de l'environnement mobile et l'application des algorithmes extraits dans cet environnement.

CHAPITRE IV

MISE EN OEUVRE

Chapitre 4: Mise en oeuvre

I.Introduction

Un réseau informatique mobile, où les algorithmes proposées peuvent être appliqués et verifiés, n'existe pas encore. Et la réalisation de ce réseau pour l'application des algorithmes s'avère impossible de part la diversité du matériel nécéssaire (MSS, mobiles, interfaces sans fil, interfaces filaires,...). Dans de telles conditions, la simulation reste notre dernier recours.

Il faudrait, donc,simuler toutes les specificités de l'informatique mobile dans un environnement distribué, pour pouvoir appliquer les algorithmes, obtenus précedement, et faire les verifications et les statistiques nécéssaires.

Dans ce qui suit, on va présenter une méthode de simulation proche du réel faite à l'université de rutgers aux étas unis, ensuite, une méthodes basée sur des tâches parallèles sous Windows95 sera discutée. Enfin, une application matérialisant cette dernière méthode sera présentée.

II.Simulation

Pour tester un protocole de communication I-TCP [Bad,97], un réseau supportant la mobilité à été réalisé à l'université de Rutgers. Ce réseau, comportant trois cellules chauvauchées, est composé d'un ensemble de stations de base et un autre de sites mobiles.

Les stations de base, utilisées dans cette expérience, sont des 486 PC-ATs 33MHZ avec 16 MO de mémoire et 400MO de disque dur, reliées entre elles par un réseau filaire Ethernet (10 Mbps).

Les hôtes mobiles utilisés sont des 486 PC-ATs 66MHZ . Les MSSs et les mobiles sont équipés de cartes NCR Wavelan 2Mbps pour supporter les communications sans fil.

Dans notre cas, puisqu' on ne dispose pas d'un environnement pareil, on va simuler sur une ensemble de processus en assurant le paralellisme entre les processus et leurs communications. Ces procesuus sont des tâches Windows95 utilisant les primitives de SendMessage et GetMessage pour communiquer. Nous avons appelé le logiciel réalisant cette simulation ***MobNet*** (***Mob***ile ***Net***work).

III.Communication

La communication entre les processus (fenêtres) dans MobNet se fait grace aux primitives *PostMessage* et *GetMessage* de Windows.

Chaque fenetre dispose d'une file FIFO (d'au maximum 100 éléments) pour reccevoir les message de windows (click sur la fenetre,redessin aprés resize,... ou parvenent des autres fenêtres)

Lorsqu'une fenêtre f1 envoie un message m à une autre fenêtre f2, en executant:

PostMessage(f2,m,Paramettres)

Windows prend le message m et ses parametttres et le dépose dans la file des messages de f2. Cette dernière retire ses message de sa file et les traite l'un après l'autre.

Pour MobNet cette dernière caractéristique est necessaire pour garantir l'exclusion mutuelle centralisée dans chaque site entre les processus demandant les ressources et les processus recevant les différents messages.

Une autre primitive qui permet l'envoie d'un message d'une fenetre à une autre est:

SendMessage(Fenetre,m,paramettres)

Cependant, cette primitive envoie le message et attend une reponse de la fenetre receptrice. C'est à dire que Windows95 suspend la fenetre emmetrice jusqu'à ce que la receptrice execute la procedure relié à ce message.

Par contre, pour la primitive PostMessage (comme l'indique son nom), Windows95 ne suspend pas la fenetre emmetrice.

IV.Paralellisme

Windows95 permet le multitaches entre les différentes applications qui se déroulent en même temps. Cependant, il ne gère pas ce mecanisme entre les fenêtres de la même application. Grace au message EV_TIMER, envoyé par Windows95 à chaque intervalle de temps à une fenêtre le demandant, MobNet, fait le déroulement des processus des différentes fenetres de façon pseudo_paralelle:

1:Fenetre1 traite le message11;

2:Fenetre2 traite le message21;

3:Fenetre3 traite le message31;

4:Fenetre2 traite le message22;

5:Fenetre3 traite le message32;

6:Fenetre4 traite le message12;

L'atomicité donc est par procedure de traitement d'un evenement.

V.MobNet

Pour simuler l'éxecution des algorithmes parallèles distribués sur différents sites, dans un environnement centralisé (WINDOWS ou UNIX), on simule ces sites par des processus parallèlles (fenêtres sous WINDOWS).

L'environnement mobile est représenté par deux types de processus: processus MSS et processus mobiles. Le premier type, simulant les stations de base, attend la reception des messages des processus du deuxieme types et il procède selon ces messages.

Le deuxieme type, simulant les stations mobiles, suit un chemin predéffinit (par l'utilisateur ou aléatoirement) et selon les actions présentées dans ce chemin il procède.

V.1.Le mobile

Une fenêtre représentant un mobile est une classe contenant les donneés suivantes, ainsi que les méthodes opérant sur ces données :

-nom du mobile,

-type de batterie du mobile,

-ensemble d'actions que doit executer le mobile,

-le mode de fonctionnement actuel du mobile (veille, normal),

-ainsi que son état (déconnecté, connecté),

A l'intialisation de cette classe, elle demande à WINDOWS de lui envoyer un messsage toutes les secondes. A l'arrivée de ce message, le mobile (fenêtre) procède selon l'action existante dans son chemin. Plusieurs actions peuvent être rencontrées:

V.1.1.Reconnection

Le mobile envoie le message RECONNECT à la station correspondante (dans le chemin) pour lui demander de l'accueillir dans sa cellule. Ce message est accompagné de son identité et, eventuellement, de l'identité de la derniére cellule qu'il a visité. La nouvelle fenêtre MSS envoie

à cette dernière un message LET avec l'identité de ce mobile pour recevoir les données qui lui sont relatifs.

Ainsi, le mobile aura changé de cellule.

V.1.2.Déconnection

Le processus de la classe mobile envoie un message DISCONNECT a la MSS courante, qui, à sa reception met à jour le champ état correspondant à ce mobile

V.1.3.Panne

Aprés l'execution de cette action le mobile n'emet ni reçoit de messages. Elle simule le cas de panne d'un mobile (endomagé ou insuffisance de l'energie disponible).

V.1.4.Out

Idem à Panne mais représente le cas où le mobile quitte la surface couverte par le réseau.

Ainsi, que d'autres actions nécéssaires pour le fonctionnement du mobile telles que la recharge de la batterie, le bascullement entre le mode veille et le mode normal.

Après l'execution de chacune de ces actions une quantité précise d'energie est consommée.

V.2.La MSS

Une fenêtre représentant une MSS est une classe contenant les informations suivantes:

-nombre de mobiles locaux,

-une liste contenant les identités de ces mobiles,

-une liste contenant leur états,

-une listes contenant les états des autres MSS (en panne ou active)

-une liste contenant les informations necessaires pour chaque mobile local.

Après son initialisation une fenêtre MSS attend la recepetion des messages des mobiles ou des autres MSS pour executer les action correspondantes.

V.3.Communication

La communication entre ces processus (fenetre) se fait sous le contrôle de WINDOWS95 par les primitives:

PostMessage: qui permet à une fenêtre d'envoyer un message à une autre, en insérent le contenu du message dans la file des messages de la fenêtre correspondante.

GetMessage: qui permet à une fenêtre de retirer un message de sa file de messages pour effectuer le traitement aproprié.

Le travail proposé permet de:

-definir un ensemble de stations de base,

-definir un ensemble de stations mobiles ,les chemins et les actions que doit suivre chacune d'elles. Et simuler dans le temps l'évolutions parallèle des mobiles, dans les cellules covrtes par les stations.

-arrêter le processus de simulation à tout instant pour visioner les informations qui nous interressent concernant les mobiles et les MSS à cet instant précis de la simulation.

-determinner à la fin les paramêtres nécéssaires pour evaluer un algorithme donné telque le nombre de messages echangés entre les mobiles et les MSS (sans fil) ou entre les MSS (filaires), le temps necessaire, la quantité d'energie consommé par chaque mobile.

-imprimer un fichier contenant l'evolution du réseau dans le temps.

Pour simuler un environnement mobile, dans le programme, on procède comme suit:

Définiton des éléments du réseau:

-nombre de cellules,

-creation des mobiles,

-définitions des chemins des mobiles (aléatoires ou prédéfinis par l'utilisateur),

Lancement de la simulation des mouvements des mobiles dans le réseau.

-création des processus des MSS,

-création des processus des mobiles,

Visualisation des résultats

-au cours de la simulation;

-ou à sa fin.

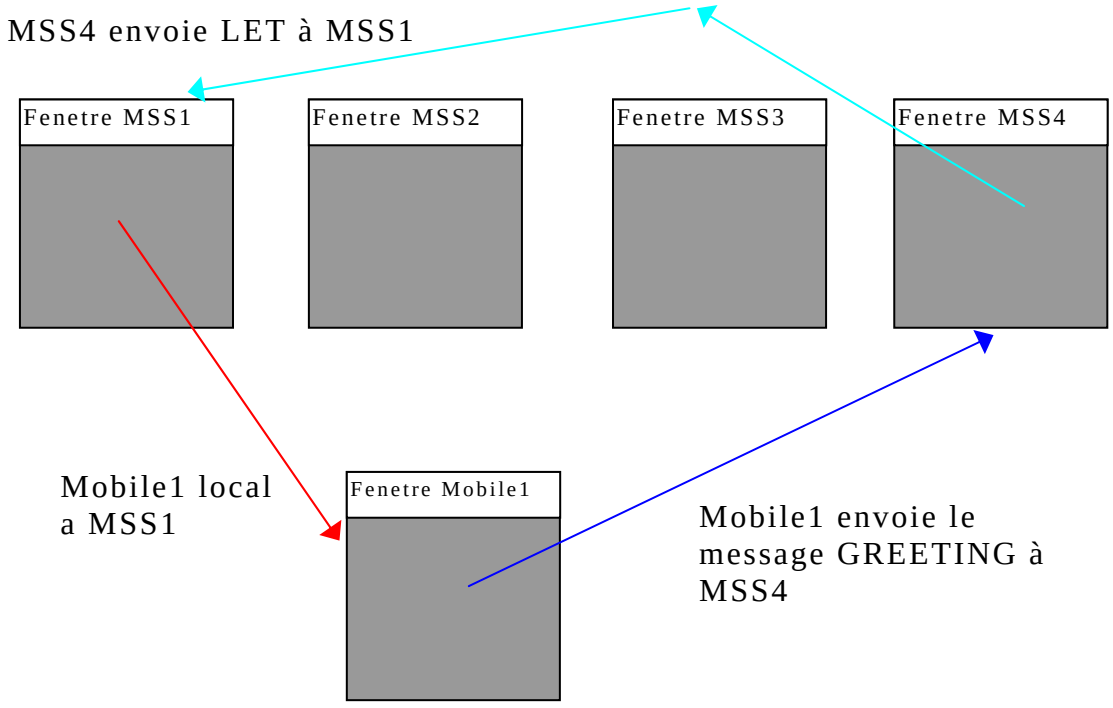


Figure 08: Représentations des différents éléments d'un réseau mobile dans MobNet.

VI.Application

Pour appliquer n'importe quel algorithme destiné à l'environnement mobile, il suffit de l'ecrire en langage C++ et d'inserer les actions qui doivent être executées par les mobiles dans la classe Tmobile et les actions relatives aux MSS dans la classe Tmss; et représenter, s'il est nécéssaire, par un dessin spécifique le résultat visualisé pendant l'execution de l'algorithme(cela dans la classe TMobileNetwork dans la fonction PaintInCell).

L'implémentation des deux algorithmes déjà présentés montre la procedure à suivre pour implémenter d'autre algorithmes.

VII.Conclusion

Dans cette dernière partie, on a présenté un moyen pour la simulation sur P.C. de l'environnement mobile (MobNet), ainsi que le principe de ce logiciel et la façon par laquelle les algorithmes de l'environnement mobile peuvent être appliqués et testés.

CONCLUSION GENERALE

CONCLUSION GENERALE

La conception des algorithmes pour les systemes distribués et leur évaluation étaient basés la supposition que la localité des différents éléments du réseau est universellement connue et ne change pas et que la connexion entre ces éléments est fiables dans l'absence des pannes.

L'introduction des hôtes mobiles invalide cette supposition et introduit de nouvelles contraintes et parametres à prendre en considération, aussi bien du point de vue des hôtes que celle de la liaison de communication.

Pour pouvoir exploiter les avantages de l'environnement resultant, il sera nécéssaire de revoir les algorithmes distribués constituant les systèmes distribués gérant ces différentes entités. Parmis ces algorithmes, celui de l'exclusion mutuelle, qui gére l'accès aux canaux de communication, et le partage des ressources informatiques classiques.

On a présenté dans ce travail un algorithme traitant chacun de ces cas, et on les a appliqué dans un environnement simulé.

L'informatique mobile fait ses premiers pas, mais déja nous pouvons voir un nombre infini d'applications qu'on pourait réaliser dans un environnement mobile.

Annexe A: *Consommation d'énergie des différents éléments d'un portable[Bag, 95]*

Périphériques	**Puissance (Watts)**
Système de base: 2 Mo , 25 Mhz	3.65
Système de base: 2 Mo , 10 Mhz	3.15
Système de base: 2 Mo , 5 Mhz	2.8
Eclairage de l'écran	1.425
Moteur du disque dur	1.1
Co-procésseur arithmétique	0.65
Lecteur de disquette	0.5
Clavier	0.49
Ecran LCD	0.315
Activités du disque dur	0.125
IC card slot	0.1
Mémoire supplémentaire (par Mo)	0.05
Port parallèle	0.035
Port série	0.03
Disque dur 1.8" PCMCIA	0.7 - 3
Téléphone cellulaire actif	5.4
Téléphone cellulaire passif	0.3
Réseau infrarouge (1 Mo/s)	0.25
Modem PCMCIA (14400 bits/s)	1.365
Modem PCMCIA (19600 bits/s)	0.625
Modem PCMCIA (2400 bits/s)	0.565
Global positioning receiver	0.67

Annexe B: Algorithmes classiques d'exclusion mutuelle basés sur l'utilisation des variables d'état.

B.1.Algorithme de la boulangerie: [Lamport74]

```
var
   choix: array[0..n-1] of booleen init false;
   numéro: array[0..n-1] of integer init 0;
 /*
   choix[i],numéro[i] sont lues et écrites par Pi, et lues seulement par Pj
,j≠i.
  */
1.choix[i]:=true;
2.numéro[i]:=1+max(numéro[0],...,numéro[n-1]);
3.choix[i]:=false;
4.pour j allant de 0 à n-1 (i≠j)
   debut
     attendre(non(choix[j]));
     attendre((numéro[j]=0) ∨ (numéro[i],i)<(numéro[j],j));
   fin.
5.Section critique

6.numéro[i]:=0;
```

B.2.Algorithme de auto-stabilisateur:[dijikstra 74]

```
   var  Drapeau:Array[1..n-1] of 0..k>1;
```

P_0	$P_{i\ (i\neq 0)}$
Attendre(Drapeau[0]= Drapeau[n-1]	Attendre(Drapeau[i]≠ Drapeau[i-1]
Section critique	Section critique
Drapeau[0]:= (Drapeau[0]+1) mod k	Drapeau[i]:= Drapeau[i-1]

B.3.Algorithme de Hehner-Symasunder:

```
Var Numéro:Array[1..n-1] of integer init +∞;

Numéro[i]:=1+Maxfini(Numéro[0], ... ,Numéro[n-1])
Pour j allant de 0 jusqu'à n-1 (i≠j)
                    attendre(Pluspetit(i,j));
Section critique

Numéro[i]:=+∞
```

Maxfini(Numéro[0],...,Numéro[n-1])=Max(Numéro[i]:Numéro[i]≠+∞)
Pluspetit(i,j)=i si (Numéro[i]<Numéro[j]) ou (Numéro[i]=Numéro[j] et i<j).

Annexe C: Algorithmes classiques d'exclusion mutuelle basés sur la communication de messages

C.1.Algorithme de Lamport:Distribution d'une file d'attente [Lamport 78]

```
Procéssus Pi
    Var     hor : integer 0..+∞
            f : array[0..n-1]of message
                  {Type message,horloge locale, numsite} init (rel,0,0)
Protocole d'accés

    diffuser(req,hor,i)
    f[i]=(req,hor,i)
    hor:=hor+1
    attendre ∀ j≠i (f[j].hor,j) > (f[i].hor,i);
```

Section critique

```
    diffuser(rel,hor,i)
    f[i]=(rel,hor,i)
    hor:=hor+1

Gestion des méssages
    A la réception de:
        - (req,h,j)
          Début
            m-à-j (hor,h)
            f[j]=(req,h,j)
            envoyer (acq,hor,i) à j
          Fin
        - (rel,h,j)
          Début
            m-à-j (hor,h)
            f[i]=(rel,h,j)
          Fin
        - (acq,h,j)
          Début
            m-à-j (hor,h)
            Si f[j].type-mess ≠ req alors f[j]:=(acq,h,j)
          Fin
        m-à-j (hor,h)
              Début
               Si hor < h alors hor:=h
               hor:=hor+1
        Fin
```

C.2.Algorithme de Ricart et Agrawala:

Processus P_i

```
Var          hor:0..+∞;
             hsn:0..+∞;
             rep:0..n-1;
             secdemandée,priorité:boolean;
             rep_différée:array[1..n]of boolean int false;
```

Protocole d'accés P_{i1}

```
secdemande:=true;
hor:=hsn+1;
rep:=n-1;
Pour j=1..n envoyer (req,hor,i) à j
Attendre (rep=0)



secdemandée:=false;
pour j=1..n faire
Debut
      Si repdifférée[j] alors
      Debut
            repdifférée[j]:=false;
            envoyer rep à j
      Fin
Fin
```

Gestion des méssages P_{i2}

```
A la réception de:

-(req,k,j)
 Debut
       hsn=max(hsn,k)
       priorité=(secdemandée)∧(k > hor)∨(k=hor)∧(i < j);
       Si priorité  alors   repdifférée[i]=true;
                   Sinon  envoyer(rep) à j;
 Fin

-(rep)
 Debut
       rep:=rep-1;
 Fin
```

Références Bibiographiques

[Bad, 93]: B.R Badrinath, T.Imielinski

Impact of mobility on distributed computation.

ACM operating system review 27 (2).

April 93.

[Bad,97]: A.V.Bakre, B.R.Badrinath

Implementation and Performance Evaluation of Indirect TCP

IEEE Transactions on computers, volume 46, numéro 3.

mars 1997.

[Bag, 95]: Aline Baggio.

Environnement mobile: Etude et synthèse bibliographique.

INRIA Rocquencourt -SOR Project, Memoire DEA,

Septembre 1995.

[Bir, 94]: K.Cho, K.P.Birman.

A group communication approach for mobile computing.

Workshop on mobile computing systems and application,

October 1994.

[IMI, 94]: T.Imielinski, B.R Badrinath

Mobile wireless computing.

Communication of the ACM, Vol 37, No 10,

October 1994.

[Sin, 95]: R.Parakash, N.G.Shivaratri, M.Singhal.

Distributed dynamic channel allocation for mobile computing.

Proceed of the 14th annual ACM symp, on principles of distributed computing (PODC 1995)

[Pss, 95]: R.Parakash, M.Singhal.

A dynamic approach to location management in mobile computing system, 1995.

[Ray, 84]:Michel Raynal.

Algorithmique de parallelisme:Le problème d'exclusion mutuelle. (DUNOD 1984).

[Ray,85]:Michel Raynal.

Algorithmes distribués et protocoles.

(EYROLLES 1985).

[Bad,93]:B.R.Badrinath, Arup Acharya et T.Imilinski.

Structuring distributed algorithms for mobile hosts.

Technical report, Rutgers DCS-TR-289/ WINLAB TR-55, April 93.

[Sin,95]:M.Singhal, R.Prakash, Department of Computer and Information Science, The Ohio State University,

N.G.Shivaratri, NEC Systems Laboratory, Inc.

A Dynamic Approach to Location Management in Mobile Computing Systems,1995.

Bibliographie

A.Acharya, B.R.Badrinath: *A Framework for Delivering Multicast Messages in Networks with Mobile Hosts,*Department of Computer Science, Rutgers University, submitted, 1994.

S.Alagar, S.Venkatesan: *Causally Ordered Message Delivery in Mobile systems,* In Proceeding of the Workshop on Mobile Computing Systems and Applications, Santa Cruz, December 1994.

A.Baggio: *Environnement Mobile: Etude et Synthèse Bibliographique,* INRIA Rocquencourt-SOR Project, septembre 1995.

A.V.Bakre, B.R.Badrinath: *Implementation and Performance Evaluation of Indirect TPC,* IEEE Transactions on computers, volume 46, numéro 3, mars 1997.

B.R.Badrinath, T.Imielinski: *Replication and Mobility,* Department of Computer Science, Rutgers University, Second IEEE Workshop on Management of Replicated Data , novembre 1992.

B.R.Badrinath, A.Acharya, T.Imielinski: *Impact of Mobility on Distributed Computations,* Department of Computer Science, Rutgers University, ACM Operating System Review, volume 27, numéro 2, avril 1993.

B.R.Badrinath, A.Acharya, T.Imielinski: *Structuring Distributed Algorithms for Mobile Hosts*, Department of Computer Science, Rutgers University, 14 *th* International Conference on Distributed Computing Systems, Poznan, Poland, mai 1994.

B.R.Badrinath, A.Acharya, T.Imielinski: *Designing Distributed Algorithms for Mobile Computing Networks*, Department of Computer Science, Rutgers University,1994.

K.Cho, K.P.Birman: *A Group Communication Approach for Mobile Computing*, Media Technology Laboratory, Canon, Department of Computer Science, Cornel University, IEEE Workshop on Mobile Computing Systems and Applications, Santa Cruz, CA, US, decembre 1994.

T.Imielinski, B.R.Badrinath: *Data Management for mobile computing,*Department of Computer Science, Rutgers University, SIGMOD RECORD, volume 22, numéro 1, mars 1993.

T.Imielinski, B.R.Badrinath: *Wireless Mobile Computing: Solutions and Challenges in Data Management*, Technical Report DCS-TR-296, Department of Computer Science, Rutgers University,1993.

T.Imielinski, B.R.Badrinath: *Wireless Mobile Computing: Challenges in Data Management*, Department of Computer Science, Rutgers University, Communication of the ACM, volume 37, numéro 10, octobre 1994.

S.Krakowiak: *Principes des Systemes d'Exploitation des Ordinateurs*, DUNOD, 1987.

A.Mukherjee, D.Siewiorek: *Mobility: A Medium for Computation, Communication, and Control*, School of Computer Science, Carnegie Mellon University.

R.Prakash, Department of Computer and Information Science,

M.Raynal, IRISA,

An Efficient Causal Ordering Algorithm for Mobile Computing Environments.

R.Prakash, M.Singhal: *A Dynamic Approach to Location Management in Mobile Computing Systems*, Department of Computer and Information Science, The Ohio State University, T-R, 1996.

R.Prakash, M.Singhal, Department of Computer and Information Science, The Ohio State University,

N.G.Shivaratri, NEC Systems Laboratory, Inc,

Distributed Dynamic Channel Allocation for Mobile Computing.

M.Raynal: *Algorithmes du Parallélisme: Probleme d'exclusion mutuelle*, DUNOD, 1984.

M.Raynal: *Algorithmes Distribués et protocoles*, EYROLLES, 1985.

M.Raynal: *Systemes Repartis et Réseaux:Concepts, Outils et Algorithmes*, EYROLLES, 1987.

Y.Slimani: *Systemes Distribués: Architectures, Concept et Algorithmes.*

A.Tanenbaum: *Réseau:Architectures, Protocoles, Applications*, InterEditions, 1992.